

# mikro számítógép magazin

Ára: 30 Ft



PÉCÉZZÜNK!

1989/7.

1088 Budapest, Rákóczi út 25.

IBM és  
Commodore

számítógépek  
javítása

# Iskolaszámítógép SZERVIZ

átalánydíjas  
szerződés

C16 bővítése 64K byte-ra  
Magyar ékezetes karakterkészlet  
beépítése

Játékprogramok vétele és  
Tectronix oszcilloszkóp eladása

előnyös  
áron!



381 121

A javítás idejére cseregépet biztosítunk!

## A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

A szerkesztőbizottság vezetője:  
Kovács Győző

A szerkesztőség munkatársai:

Bakos Tamás  
(programozástechnika)  
Broczkó Péter  
(hírek)

Kovács Győző  
(levelezés)

Nagy Imre  
(tanuljuk együtt)

Petróczy Judit  
(könyvek)

Pinke György  
(Enterprise)

Soltészné Vizi Zsuzsa  
(tervezőszerkesztő)

Simonyi Endre

Szebenszki Sándor

Szulovszky Csaba

Tamásné Lakó Erika

Terebessy Ákosné

Varga János

**Cimképünk:**  
Velekey József Lajos  
munkája




Felelős szerkesztő:  
Könyves Tóth Pál

Szerkesztőség:  
1027 Budapest, Fő u. 68.  
Telefon: 154-250

Levél cím:  
1371 Budapest  
Pf. 433

Kiadja:  
MTESZ Neumann János  
Számítógéptudományi Társaság  
1054 Budapest, Báthory u. 16.

Levél cím:  
1368 Budapest 5. Pf. 240

Telefon: 329-349

Felelős kiadó:  
Tóth Istvánné ügyvezető  
főtitkár/helyettes

Terjeszti a Magyar Posta  
Előfizethető a hírlapkézbesítő  
hivataloknál  
és a Posta Hírlap-előfizetési  
és Lapellátási Irodáján  
(1900 Budapest XIII.,  
Lehel u. 10/A)  
vagy átutalással a 215-96 162  
pénzforgalmi jelzőszámra.

Megjelenik havonta.  
Egy szám ára 30,— Ft  
Előfizetési díj:  
egy évre 360,— Ft  
fél évre 180,— Ft  
Külföldön terjeszti  
a Kultúra,  
1389 Budapest, Pf. 149  
és a Magyar Média  
1932 Budapest, Pf. 279  
88—1135



Szikra Lapyomda  
Budapest (89—0911)  
Felelős vezető:  
Csöndes Zoltán vezérigazgató

INDEX: 25 629  
ISSN 0236-6088

### TARTALOM

- |    |   |
|----|---|
| 2  | Tanulási kultúra, tanulási technológia                    |
| 9  | STARTOOL—EDASS — a legjobb?!                              |
| 11 | Feladatok — megoldások                                    |
| 12 | Nemes Tihamér Országos Középiskolai<br>Tanulmányi Verseny |
| 27 | Rendszerfejlesztési eszközök                              |
| 38 | Párhuzamos interfész                                      |
| 41 | Az AmigaDOS   |
| 45 | Programtermék — Vegyészeti csemegék                       |
| 47 | Adok-veszek-cserélek                                      |

### TANULJUK EGYÜTT!

3

- |   |                          |
|---|--------------------------|
| 3 | A Pascal rejtelmei       |
| 5 | Szociometriai „gyorsító” |

### CSIPEGETŐ

14

- |    |                                   |
|----|-----------------------------------|
| 14 | Action Replay MK V                |
| 14 | Módosított másolás                |
| 15 | Nem ördögösség!                   |
| 15 | Egyből négyet                     |
| 15 | Miből lesz a cserebogár?          |
| 16 | Programozási stílusok különkiadás |
| 16 | TOP-lista                         |

### PROGRAMOZÁSTECHNIKA

17

- |    |                                      |
|----|--------------------------------------|
| 17 | Konkurens programozás                |
| 20 | Rezidens óra                         |
| 22 | Programozási fogások és melléfogások |

### ENTERPRISE

23

- |    |                                   |
|----|-----------------------------------|
| 23 | Hívja önmagát!                    |
| 24 | Kalandozások a karaktermemóriában |
| 25 | Enterprise-TOTÓ 1989              |
| 25 | Megváltozott karakterek           |
| 26 | Egy kis botkormányosdi            |
| 26 | Mi a manó?                        |

### PÉCÉZZÜNK!

30

- |    |   |
|----|---|
| 30 | Norton Editor                                     |
| 32 | A PKARC tömörítőprogram                           |
| 35 | Nyolc be- és nyolc kimenetes kártya IBM PC géphez |

### μKLUB

43

- |    |                      |
|----|----------------------|
| 43 | Adom a magyarázatot! |
|----|----------------------|

### SAKK

44

- |    |                              |
|----|------------------------------|
| 44 | Figuratípusok jutalompontjai |
|----|------------------------------|

### KÖNYVEK — HÍREK — ÉRDEKESSÉGEK

46

### AZ OLVASÓ ÍRJA

48

„Harun ar-Rasid és al-Mámun alatt kialakított kultúra egy évezredre meghatározta Elő-Ázsia képét. (...) Al-Mámun alatt állami támogatást kaptak a tudományok és a művészetek. A kalifa személyesen foglalkozott az ókori természettudósok műveivel, ő ösztönözte arab fordításukat. A „Tudományok háza”-ban könyvtárat alapított és tudósokat foglalkoztatott. Damaszkuszban és Bagdadban csillagvizsgálók működtek. (...) Jelentőset alkotott az indiai és görög tudásra építő abházda matematikusok. Közülük az egyik legismertebb a 820 körül alkotó Muhammad al-Horezmi algebráról szóló könyve Európában is elterjedt. Nevéből származik az algoritmus fogalma.”  
(Buchard Brentjes: Izmael fia.)

# Tanulási kultúra, tanulási technológia

Új munkakörömből adódóan az utóbbi időben sokat utazom Keletre és Nyugatra egyaránt. Sok szakmabelivel van alkalmam találkozni a kezdő szakembertől egészen az akadémikusig, tudóssal, tudománytervezőkkel, pedagógusokkal, üzletemberekkel, diákokkal, társadalmi munkásokkal. A paletta meglehetősen színes. A mai Magyarországról jövők elbeszéléseit Keleten is és Nyugaton is egyre fokozódó és a világ mindkét részén velünk szimpatizáló érdeklődés kíséri. Elég csak felidézmem a londoni European Business Schoolban tartott beszámolómat és az azt követő beszélgetéseket az iskola hallgatóival, vagy a bulgáriai matematikai kongresszuson résztvevő középiskolásokkal eltöltött estét, de nem volt rossz a párizsi UNESCO palotában élvezett kérdés-„pergőtűz” sem. Persze én is kibaszálтам az alkalmat, és hasonló módon megpróbáltam információit gyűjteni elsősorban arról, hogyan is áll az informatika és az iskola kapcsolata, mennyire használják fel a közoktatásban, az egyetemi-főiskolai és a szakoktatásban, illetve a felnőttoktatásban a számítógépeket. Megpróbáltam én is elmondani, hogy milyen tervek foglalkoztatják és a válaszokból lemerni, érdemes-e hosszú távon a távtanulással és a számítógéppel támogatott tanulást foglalkozni.

Elojáróban talán azt kell elmondanom, hogy a számítógéppel való tanulás – elsősorban Európában, azon belül is főleg a szocialista országokban – még nem mondható egyetelműen sikeresnek. Ez persze nem azt jelenti, hogy az illetékesek nem minősítik annak, hanem sokkal inkább azt, hogy a számítógép ma még nem tudott az iskolában „nélkülözhetetlen” tanulási eszközzé válni. Sokkal beszéltem, mindenkinek más volt a véleménye, hogy miért nem. Egy a problémát nálunk abban látom, hogy a szocialista országokban még mindig megvan a központi oktatásirányítás. Az ember azt hinné, hogy egy centralizált rendszerben könnyen lehet a meglévő anyagi erőket egy jó cél érdekében koncentrálni, ami biztosan igaz, de persze az is, hogy a piramis csúcásán hozott rossz döntés az egész rendszer működését megbénítja. Persze vannak a vártnál kisebb eredmények más okai is; például a Szovjetunióban, de Bulgáriában is. Megpróbáltam ugyanis óriási befektetéseket elfogadható árú iskolai számítógépeket előállítani, ami sajnos nem sikerült, mert a kis szériában, manuálisul készülő termékek között gyártott gépek megbízhatósága nagyon alacsony, az ára pedig túl magas, a megfelelő olcsó tajvani vagy dél-koreai termékekhez viszonyítva. Nálunk más a helyzet, mi a sokféle 8 bites gép terhért nyögünk, amelyekkel alig tud kezdeni valakit is az iskola. Erről sokszor, sokat és általában hiába írtam, a helyzet gyakorlatilag nem változott.

A másik, világszerte meglévő probléma, hogy tanítsunk-e informatikát, vagy ne tanítsunk; ha igen, akkor minden diáknak, vagy csak a kiválasztottaknak, és főleg az, hogy mit. A Szovjetunióban pl. még mindig él az elmélet, hogy informatikát lehet számítógép nélkül is tanítani. Erre az elméletre azért volt szükség, mert nem volt elég számítógép

az iskolákban, de azért a tanárok szerettek volna lépést tartani a fejlődéssel. Ők is vettek japán 8 bites gépeket, szoftver nélkül, és igen tiszteletreméltó erőfeszítéssel megpróbálták a gépekből valamit kicsihiolni. Így aztán ők se jutottak tovább, mint mi, az informatika – programozás képlettel gyötrik a diákok nagy részét. Bolgár diábarátaim egy számítástechnikai szakközépiskolából jöttek, valamennyien megszállott programozók voltak, így számukra az informatika – programozás képlet egyértelmű volt. Arra a kérdésre pedig, hogy mi történik azzal a gyerekkel, aki nem akar programozni, „csak” használni szeretné a számítógépet, vállonogatás volt a válasz. Szerencsére nálunk ilyen kötelező diszciplína az iskolákban nincs, a számítógéppel való ismerkedés főleg szakkörökben, illetve matematika- vagy technikaórákon folyik.

Azt hiszem, hogy megköczkíthatom azt az állítást: a fő probléma – elsősorban Európában – az, hogy nincsenek jó tanulóprogramok. Nem rossz a helyzet pl. matematikában vagy a nyelvi tanulásban, vannak fizikai-kémiai kísérleteket szimuláló programok is, amelyekről részben előadásokon hallottam, részben pedig láttam is jó néhányat. Pedig néhány országban igazán nem lehet azt mondani, hogy a kormányok viselkednek al-Mámun uralkodóhoz hasonlóan, hiszen tekintélyes összegeket fektetnek a számítógépes tanulóprogramok fejlesztésébe, még sincs elég, és ami van, az sem terjed megfelelő mértékben. Nem alakult ki ugyanis sem iskolák közötti sem nemzetközi együttműködés az elkészült tanulóprogramok kölcsönös használatára. Az UNESCO konferencián az egyik igen tekintélyes előadó azt bizonygatta, hogy nehez az a országban készült tanulóprogramot B országban bevezetni, mert mások a körülmények, az oktatási hagyományok, a diákokkal szemben támasztott követelmények. Véleményem szerint nem elég egy tanulóprogramot az A ország nyelvéről a B-re lefordítani, azt át is kell dolgozni, illeszteni kell a B országbeli körülményekhez. Véleményem szerint ez nincs egészen így, amit az előadás után az előadónak el is mondtam. Pontosabban elhiszem az állítást néhány, az adott ország viszonyaihoz szorosan kötődő tantárgy esetében, mint pl. háztartásgazdaság vagy történelem, esetleg irodalom, de általában nem értek vele egyet. Az viszont egészen biztos, hogy a tanárok – ezt többen is mondták – nehezen fogadják el egy másik pedagógus tanulóprogramját, különösen nehezen fogadják el akkor, ha azt nem egy pedagógus, hanem pl. egy kevés pedagógiai érzékkel megáltdott számítógépes programozó készítette. A pedagógus, mint a tankönyvet, szeretné a neki ajánlott tanulóprogramot módosítani, kihúzni belőle, beletoldozni valamit, hogy a diák a tantárgyat úgy ismerje és tanulja meg, ahogyan azt ő szeretné.

Márpedig pl. egy latintanárral valószínűleg képtelen arra, hogy BASIC-ben a latin nyelv tanulására írjon programba belenyúljon, és azt a saját elképzelései szerint az eredeti programozó segítség nélkül átalkítsa. Erre csak akkor van módja, ha a számítógépes ta-

nulási anyagot a szerző valamelyik magas szintű szerzői rendszer és nem egy számítógépes nyelv segítségével készítette. A szerzői rendszer az a technológia, ami nélkül ma már nincs, illetve csak nagyon kevés nemzetközi szinten is elfogadott produktum van.

Sajnos vagy talán hál’ Istennek ma a technológiát is elárasztanak bennünket, nagyon nehéz jól választani. A legegyszerűbb persze becsukni a szemünket, és hagyni menni a dolgokat a maguk útján, irti az alig használható tanulóprogramokat és készíteni a csodálatos statisztikákat, hogy milyen jól is megy nálunk az iskolai számítástechnika. Nehezebb kiválasztani és bevezetni egy olyan technológiát, amely nemcsak a hazai, de a nemzetközi tapasztalatserét és termékszerét is biztosítja. Az UNESCO-konferenciák egyik szekciójában ezt el is mondtam, amikor az egyik fejlődő országból küldött arról panaszokdott, hogy kevés tanulóprogramot tud átvenni, mert náluk a diákok nem rendelkeznek elegendő alapképzéssel ahhoz, hogy pl. a francia oktatásban használt tananyagokat megértse. Elmondtam, hogy szerintem „fordítva ültek a lóra”, előbb a technológia, utána a termék. Senki sem kötelez arra senkit, hogy valamilyen változtatás nélkül alkalmazzon, de butaság volna mindent az alapoktól kezdve kifejleszteni, ha mások eredményeit akár részben is fel tudják használni.

Véleményem szerint a legfontosabb feladat számunkra is, hogy találjunk olyan tanulási anyag-bankot, ami tartalmilag számunkra elfogadható, végük meg ezeket a termékeket, de technológiától, és terjesszük el az egész országban a használatát. Ha például informatikában gondolkodom, akkor itt van a szomszédban a COSTOC tanulásianyag-bank (és a hozzávaló AUTOOL vagy HyperTRAIN szerzői rendszerek, amelyek egymással kompatibilisek), amelyet a Grazi Egyetem H. Maurer professzor vezetésével és még további 18 külföldi egyetem, főiskola, kutató- és fejlesztőintézet közreműködésével fejlesztettek ki. Ezzel az anyaggal, ami ráadásul magyar VTX-rendszeren is futtatható, nagyon rövid idő alatt el lehet kezdeni egy távtanulási fősiskola megszervezését, nem beszélve arról az előnyről, hogy a nálunk ugyanezen a technológiával fejlesztett tananyagokat a COSTOC-ot használó külföldi tanintézeteknél is értékesíteni lehet.

Még egy utolsó megjegyzés, mert erről is szó volt a már említett konferencián és megbeszéléseken. Ellenében az itthon elterjedt nézetekkel, a jó tanulási anyag kifejlesztésébe pénzbe kerül, nem is kevésbe, sokba. De megéri, és főleg akkor éri meg, ha ezeket az anyagokat nem a hagyományos, hanem a tömegközvetítésben, távtanulási rendszerekben alkalmazzák. Azt nagyjából tudni lehet, hogy ma az oktatási kormányzat pénzügyi helyzete nem hasonlít össze Harun ar-Rasid és al-Mámun uralkodók lehetőségeivel, de talán nem üres még annyira az a zseb, hogy ne lenne érdemes erre a célra belenyúlni, és akkor mi is meghatározhatnánk legalább egy Abházda országunk kulturális és tudományos képét.

Kovács Gyöző



# A PASCAL REJTELMEI

## 11. Fájelkezelés

A Pascal egyik sajátossága, hogy a bevitelt és a kivitelt szolgáló eszközöket (perifériákat) egységesen kezeli, eltérően azok fizikai működésének sokféleségétől. Ezen eszközök összefoglaló neve állomány (file, „magyaritva”: fájl).

Bevitellel és kivitellel már eddig is foglalkoztunk, de az alkalmazott **write**, **writeln**, **read** és **readln** eljárások használatánál elhallgattuk, hogy tulajdonképpen fájlok kezeléséről van szó. A magyarázat igen egyszerű: a Pascal két perifériát, a billentyűzetet és a képernyőt kitüntetetten kezeli, az eljárások hívásakor ezek fájlneveit nem kell megadni, pontosabban: ha fájlnevet nem adunk meg, a bevitelt a billentyűzetre, a kivitelt a képernyőre vonatkozik. (Egyébként a két perifériának, mint fájlnak a neve: **input** és **output**.) A billentyűzet nem normál — echo nélküli — használatukhoz viszont a **kbd** fájlnevet kell megadni, mint ahogyan ez a már elkészült programjainkban is előfordult.

A különböző Pascal-reprezentációk (és ez a Turbo Pascalnál is így van) kétféle fájl típust is alkalmazhatnak. Az egyik — amelyik a programozásban már jartasak számára ismerős lemezes adattárolományoknak felel meg — a **file**-adattípus. Ez elvileg végtelen mennyiségű, azonos típusú adat összességét jelenti. Az adatok típusára csak egyetlen korlátozás van: egy fájl eleme újabb fájl nem lehet. Deklarációja a következő:

**var változónev: file of adattípus;**

Szerkezete egységes, a fájl elemei nem alkotnak egymástól elkülönülő kisebb-nagyobb csoportokat (sor, rekord stb.). A fájl elemeinek elválasztására a fájlmuveleteket előíró eljárásokban a **(vevőszó)** karakteret szolgál (mint ahogyan ezt a **write** és a **writeln** eljárásokban is alkalmaztuk). A fájl végét a lemezre írt fájlvégjel, a **Ctrl-Z** karakter jelzi. Ennek neve a Pascalban **eof** (end of file).

A másik fájltypus elsősorban szöveges állományok „tárolására” alkalmas. Nemcsak lemezmuveleteknél, hanem bármilyen periférián végzett be- vagy kivitelnél is használható.

Deklarációja:  
**type text = file of char;**

A típusdeklarációra egyébként nincs szükség, mert a **text** előre definiált típus, és így a Turbo Pascal felismeri, azaz a

**text** kulcsszó változódeklarációkban közvetlenül alkalmazható. Például a **type text = file of char;**  
**var szöveg: text;**  
helyett elegendő csak a változódeklaráció:

**var szöveg: text;**

Eddigi programjainkban tulajdonképpen a **text** típusú állományokkal végeztünk perifériamuveleteket (**read**, **readln**, **write**, **writeln**).

### 11.1 A file adattípus használata

A file adattípust elsősorban lemezes állományok létrehozásakor, illetve kezelésekor alkalmazzuk. A felhasználást számunkra a Turbo Pascalra gondolunk — beépített eljárás és függvény támogatja.

Egy fájl használatát mindig a logikai fájl, azaz a programban használt fájlnev és a fizikai fájl, azaz a lemezen lévő fájl nevének egymáshoz rendelésével kell kezdeni. Erre az **assign** eljárás szolgál, amelynek szintaxisa a következő:

**assign (fájlnev, sztring);**  
ahol a fájlnev a programban használt változónev, a sztring a fizikai fájl neve. Például:

```
assign (fájl, 'nevsor.dta');  
assign (adatok, 'b: \pasdata\dat.bdf');
```

A példák a **fájl** és a **nevsor.dta**, illetve az **adatok** és a **data.bdf** logikai és fizikai fájlokat rendelik egymáshoz. A második példában a fizikai fájlnev előtt az aktuális lemezmeghajtó egységre és az aktuális könyvtárra vonatkozó utalás van. Természetesen ilyen hivatkozásokor a könyvtárnak a lemezen léteznie kell. Ha a Pascal-rendszer és a fizikai fájlok ugyanazon a lemezen és ugyanabban a könyvtárban találhatók, az ilyen megjelölés nem hiba, de nincs szükség rá. Az összerendelés minden más fájlmuvelet előtt kell elvégezni, másképp a Pascal és a fizikai fájl kezelő DOS együttműködésére nincs lehetőség. Ekkor a program futás közben I/O hibajelzés: **file disappeared** (a fájl eltűnt) kíséretében leáll.

Új fájl megnyitását írásra a **rewrite** eljárással végezhetjük el, ez egyébként az

azonos néven nyilvántartott fájl törlési is. Ezért új fájlok létrehozása előtt célszerű a meglévő állományokat megvizsgálni, például a főmenüben lévő **D** paranccsal. Az eljárás szintaxisa:

**rewrite (fájlnev);**

A létrehozott fájl a megnyitás után üres, elemeket nem tartalmaz. A megnyitás után a fájlba a **write** eljárással írhatunk:

**write (fájlnev, változó1, változó2, ...);**

Természetesen a fájlba írt változóknak a fájldeklarációban megjelölt adattípussal (a fájl alaptípusával) egyezniük kell.

Egy már meglévő fájl megnyitását a **reset** eljárással végezhetjük el:

**reset (fájlnev);**

Ezután a

**read (fájlnev, változó1, változó2, ...);**

eljárással a fájl alaptípusával megegyező változó1, változó2 stb. változóba beolvasásra kerülnek a fájl egymást követő elemei. A **reset**-tel megnyitott fájlba a **write** eljárással, a már megismert módon új adatokat is írhatunk, ezek az eredeti fájl végéhez kapcsolódnak. Használat után a fájlok a

**close (fájlnev);**

eljárással zárhatók le.

A nyitott fájloknak az eddigiek alapján csak sorrendi (szekvenciális) hozzáférés valósítható meg. Ez azt jelenti, hogy a fájl egyes elemeihez csak sorrendben, például olvasásnál a felírási sorrendjével megegyező módon férhetünk hozzá. A fájl tevékeny elemének megkereséséhez a Pascal a **seek** eljárást kínálja. A

**seek (fájlnev, elemszám);**

alkalmazásával az adott sorszámú elemre pozícionálhatunk, azaz bármelyik elemhez közvetlenül hozzáférhetünk.

Az aktuális elemszám lekérdezése a **filepos (fájlnev)**

függvény segítségével végezhető el. Ennek különösen akkor vehetjük hasznát, ha az aktuális elemhez képest a fájlban, „előre” vagy „vissza” kívánunk lépni meghatározott számú elemmel. A következő példa öt elemmel való visszalépést szemlélte:

**a = filepos (fájlnev);**

**seek (fájlnev, a-5);**

vagy egyszerűbben:

**seek (fájlnev, (filepos (fájlnev))-5);**

A szekvenciálisról eltérő hozzáférés a program futási idejét lényegesen csökkenti, ha egy fájlban az egyes elemeket „összevissza” többször kívánjuk elérni. (Ha csak sorrendi hozzáférésre lenne



mód, a fájlt mindig a kezdetétől kellene újraolvasni.)

A programozási kényelmet szolgálják a Pascal további fájlkezelő eljárásai és függvényei is.

A fájl mérete — a fájlban szereplő elemek száma — a **filesize** függvénnyel kérdezhető le a következő formában:

**a:=filesize (fájlnev);**  
Ha a fájl üres, a=0 értéket kapunk.  
A fájl végének megfigyelésére az **eof** függvény alkalmazható. Értéke **true**, ha az olvasott elem a fájlvég-jel (a Ctrl-Z karakter). E lehetőség használatát szemlélteti a következő programrészlet:

```
repeat
  read (fájlnev,a);
until eof (fájlnev);
```

A fájlból a kiolvasott adatok az a változóba kerülnek; az olvasás a fájlvég-jelig folytatódik.

Egy lezárt fájl törlését az **erase (fájlnev);**

eljárással érhetjük el.  
Végezetül a **36. ábrán** — a magyarázatot a komment sorokra bízva — egy programot mutatunk be, amely a fájlkezeléssel kapcsolatban ismertett minden lehetőséget felhasznál.

## 11.2 A text adattípus használata

Mint ahogyan azt már a fájlokról szóló bevezetőben említettük, a **text** előre definiált **file of char** típus. A **file** típusú képest lényeges eltérés, hogy a fájl elemei, a karakterek nem egyetlen formázatlan halmazt képeznek, hanem különböző hosszúságú sorokra töröltek CR/LF (**eoln**: end of line) sorvégejelekkel. A fájl szerkezetéből adódik az is, hogy a **seek**, **filesize** és a **filepos** eljárások a fájlműveletek során nem használhatók.

Egy fájl megnyitása a **rewrite (fájlnev)** vagy a **reset (fájlnev)** eljárással végezhető el. A **rewrite** után csak írni, a **reset** után csak olvasni lehet. Már létező fájl folytatódagos írásához a fájl **append (fájlnev)**-vel kell megnyitni.

A **readln** a következő sor elejére pozícionál, a **writeln** sorvégejelet ír a fájlba.

Az **eoln (fájlnev)** értéke **true**, ha a sor végére (a CR/LF utánra) pozícionáltunk, és ha a fájl végén, a Ctrl-Z karakter után vagyunk. Az **eoln (fájlnev)** értéke **true**, ha a fájl végére pozícionáltunk.

A **read (fájlnev, változó1, változó2,...)** eljárással a változó1, változó2,... változóba olvashatunk értékeket. Változók lehetnek:

- **char**: a következő karakter (a sor végén CR/LF);
- **string**: a maximális (255) számú karakter, de az olvasás legfeljebb a sorvégeig vagy a fájlvégig tart;

```
program fajl;
var elemsorszam,hossz,ujadat,
    f,g,h,fi,gl,hl,keresett_elem:integer;
    ch:char;
    fajl:file of integer;
begin
  clrscr;
  writeln('A fájlba írandó integer adatok:');
  writeln;
  write('f=');readln(f);
  write('g=');readln(g);
  write('h=');readln(h);
  assign(fajl,'a:\pascfile\fajl.prb');
  {a logikai és a fizikai fájl egymáshoz rendelése}
  rewrite(fajl);
  {a fájl megnyitása írásra, a régi - ha van - törlődik}
  write(fajl,f,g,h);{írás a fájlba}
  writeln;writeln('A fájl felírása a lemezre');
  close(fajl);
  {a fájl lezárása}
  writeln('Folytatás bármely billentyű');
  read(kbd,ch);
  clrscr;
  writeln('A fájl tartalmának olvasása a lemezről');
  reset(fajl);
  {a létező fájl megnyitása}
  read(fajl,fi,gl,hl);
  {olvasás a fájlból}
  writeln;
  writeln('A fájlból olvasott adatok:');
  writeln('fi=',fi);
  writeln('gl=',gl);
  writeln('hl=',hl);
  writeln;
  writeln('Új adat hozzáírása a fájlhoz');
  write('Az új integer adat: ');
  readln(ujadat);
  write(fajl,ujadat);
  close(fajl);
  writeln('Folytatás bármely billentyű');
  read(kbd,ch);
  clrscr;
  writeln('Elemek keresése a fájlból');
  writeln;writeln('Az elemszám olvasása a fájlból');
  reset(fajl);
  repeat
    writeln;write('Az elem sorszáma (0...');
    hossz:=filesize(fajl)-1;
  {a fájl elemének száma=az utolsó elem sorszáma}
  write(hossz,'? ');
  readln(elemsorszam);
  seek(fajl,elemsorszam);
  {a kívánt elem keresése}
  read(fajl,keresett_elem);
  {a kívánt elem kiolvasása}
  writeln;writeln('A keresett fájlilelem tartalma:');
  writeln(keresett_elem);
  writeln('Folytatás bármely billentyű, Vége:ESC');
  read(kbd,ch);
  until ch=#27;
  close(fajl);
```

```
clrscr;
writeln('Olvasás a fájlvég jeléig');
writeln;
reset(fajl);
while not eof(fajl) do
  {a fájlvég jeléig olvas a fájlból}
  begin
    write('A kiolvasott elem sorszáma és tartalma: ');
    write(filepos(fajl),' ');
    {kiírja az éppen mutatott (most a 0.-dik) elem pozícióját}
    seek(fajl,filepos(fajl));
    {a mutatott elemet keresi}
    read(fajl,f);
    {olvas, egyben pozícionál a következő elemre}
    writeln(f:7);
  end;
  close(fajl);
  erase(fajl)
  {törli a fájlt}
end.
```

36. ábra

```
program textfile;
var ch:char;
    szoveg:string[255];
    textfajl:text;
begin
  assign(textfajl,'a:\pascfile\fajl.txt');
  rewrite(textfajl);
  {rewrite után csak írni lehet a fájlba}
  repeat
    readln(szoveg);
    writeln(textfajl,szoveg);
    writeln('Vége:ESC, Folytatás: minden más billentyű');
    read(kbd,ch);
  until ch=#27;
  close(textfajl);
  writeln('A fájl felírása a lemezre kész');
  writeln('A kiíráshoz nyomj le egy billentyűt!');
  read(kbd,ch);
  reset(textfajl);
  {reset után csak olvasni lehet a fájlból}
  repeat
    readln(textfajl,szoveg);
    writeln(szoveg);
  until eof(textfajl);
  close(textfajl);
end.
```

37. ábra



— **integer** vagy **real**: az olvasás szövegig vagy fájlvégig tart, az eljárás az adat típusának megfelelően a karaktersorozatot numerikusra konvertálja.

A **readin** eljárás a **read**-től csak abban tér el, hogy a sor végére pozícionál. Ha tehát kevesebb adatot olvasunk ki a **read**-ban felsorolt változóba mint amennyi a sorban van, a hátralévő karakterek elvesznek.

A **write** (fájlnev, kifejezés1, kifejezés2, ...) eljárással a kifejezés1, kifejezés2, ... kifejezéseket írhatjuk a fájlba. Kifejezések lehetnek:

- **char** [:n]: karakter kiírása n helyre, jobbra igazítva, alaphelyzetben  $n=1$ ;
- **string** [:n]: sztring kiírása n helyre, jobbra igazítva, alaphelyzetben  $n=$  a deklarált maximális hossz;
- **boolean** [:n]: **true** vagy **false** kiírása n helyre, jobbra igazítva, alaphelyzetben  $n=4$  vagy  $n=5$ ;
- **integer** [:n]: egész kiírása n helyre, jobbra igazítva;
- **real** [:n:m]: valós kiírása n helyre, m tizedessel, jobbra igazítva, alaphelyzetben  $n=18$ ,  $m=10$ .

A **writeln** eljárás a **write**-től csak abban tér el, hogy a sor végére sorvégejel (CR/LF) ír. A kifejezések típusának felsorolása után szögletes zárójelbe írt paramétereket találunk. Ezekkel a 9.1 formázott kivitel című részben már találkoztunk,

igy alkalmazásukról most nem szükséges ismét szólni.

Egy szövegfájl létrehozására és kezelésére mutat példát a 37. ábrán látható program. Hozzá részletes magyarázatot külön nem fűzünk, a komment sorokban működéséről megfelelő mennyiségű információ található. A program a fájlba íráskor meglehetősen bonyolultnak tűnik; ennek oka, hogy a sorok száma előre ismeretlen, így a **repeat...until** típusú ciklust kellett alkalmazni. Ha a sorok számát ismerjük, a **for...to...do** ciklussal egyszerűbb szerkezetet valósíthat meg.

Mint a 11. részben már említettük, a Pascal számára bármilyen be- és kivitel egy fájlból való olvasást vagy egy fájlba való írást jelent. Ha tehát valamelyik perifériával kommunikálni akarunk — nem a file adattípussal dolgozunk —, a perifériát egy változódeklarációban egy **text** állományként kell meghatározni. Például:

**var disk, printer, keyboard: text;**

A billentyűzetet és a képernyőt minden Pascal-rendszer, így a Turbo Pascal is kintüntetetten kezeli. A billentyűzet az **input**, a képernyő az **output** előre definiált állomány, ezeket a fájlneveket tehát a **read**, **readin**, **write**, **writeln** eljárásokhoz tartozó változó vagy kifejezés listákban nem kell feltüntetni. (Ezt egyébként — bár akkor „szakszerű” magyarázat nélkül — a 3. részben már megemlítettük.) Ha már a perifériákról, mint állományokról

szó esett, felsoroljuk a Turbo Pascalban használható, a lényeges perifériákhoz tartozó logikai fájlok neveit és az adathírnyit:

con: konzol	bevitel, kivitel
trm: terminál	bevitel, kivitel
kbd: billentyűzet	bevitel echo nélkül
lst: nyomtató	kivitel

Ezeken kívül a konzolhoz és a terminálhoz alaphelyzetben további fájlok vannak hozzárendelve:

input: bevitel  
output: kivitel

Ezek jelentik a már többször említett fájlneveket (billentyűzet: bevitel, képernyő: kivitel). Ez a hozzárendelés programból meg is szüntethető. Ezzel a kérdéssel a 13. részben, a futtatási opciók tárgyalása során részletesebben is foglalkozunk.

A con és a trm készülékeknél a sor olvasása pufferbe történik, amelynek tartalma a sor Enterral lezárása előtt javítható, például a Backspace vagy a Del billentyűkkel. A puffer hossza alaphelyzetben 127 bájtt, amely azonban a **buflen** nevű, előre definiált változó felülírásával (új érték adásával) megváltoztatható. A változtetés csak egy, a pufferből való olvasás (**read** vagy **readin**) tartamára érvényes, utána az eredeti érték áll vissza. A beolvasott fájl vége a Ctrl-Z-el jelezhető. A kbd-n a programban előírt számú karakter beolvasása után az olvasás véget ér.

Nagy Imre

## Közösségek szerkezetének vizsgálata

# Szociometriai „gyorsító”

A lelkiismeretes, hivatását és tanítványait szerető pedagógus sok egyéb feladata mellett nem tud elegendő időt fordítani arra, hogy osztályát jobban megismerje. A többség leginkább intuícióval gyűjtött információmorzsákra hagyatkozik a rábizott közösség megítélésében, formálásában. Nem térhetünk itt ki annak a bő eszköztárnak az ismertetésére, melyet az egyének és közösségek egzaktt vizsgálatára dolgozott ki a pedagógia. Nincs is erre szükség, hiszen a tanulmányi idejében minden pedagógus megismerkedhetett ezekkel, a mellőzősüknek viszont az egyik legfontosabb oka éppen a krónikus időhiány.

A számítógépek elterjedésével azonban olyan eszköz került a gyakorlati pedagógus kezébe, amely az időigényes pszichológiai és szociometriai vizsgálatok elvégzését rendkívül megkönnyíti. A vizsgálati módszerek nagy

többsége igen egyszerűen gépesíthető. Itt egy olyan vizsgálatot mutatunk be, melynek programja nem tartozik az egyszerűek közé. Közreadásával nemcsak az a célunk, hogy közkinccsé tegyük, hanem arra is szeretnénk biztatani a számítástechnikával foglalkozó tanárokat, hogy ilyen vagy ennél is egyszerűbb programokkal segítsék más szakos társaikat.

### Szociometriai kérdőív

A közösség szerkezetének vizsgálatára alkalmazott módszerek egyike a következő. Minden tanulóknak néhány olyan kérdést teszünk fel, amelyre egy vagy több társát megnevezve kell válaszolnia. A kérdések az életkortól függően valamilyen valóságos vagy elképzelt szituációt írnak le. Például: Ki a legjobb fej az osztályban?

vagy: Kivel élne legszívesebben egy lakatlan szigeten? stb. A válaszok a kérdés jellegétől függően szimpátia- vagy antipátia-szavazatok képviselnek. A megbízható kiértékelés érdekében a kérdőív elegendő számú személy tartalmaz, és kérdésenként több személy megjelölését írja elő. Ez az oka annak, hogy a vizsgálat kiértékelése nagyobb létszámú csoportnál igen sok időt igényel.

A szakirodalomból ismert szociometriai modellek koncepciójának az a jellegzetessége, hogy a módszerek a manuális feldolgozást feltételezve alakultak ki. Ezért a nagy mennyiségű vagy bonyolultabb számítást igénylő eljárásokat igyekeztek mellőzni. A számítógép alkalmazása ezeket a gondokat megszünteti, sőt lehetőséget teremt arra is, hogy mélyebb betekintést engedő modelleket is bevonjunk a strukturális vizsálatba. A felhasználható modellek kö-



rét bővítik többek között azok a gráfelméleti alkalmazások, melyeknek a kivitelezése számítógép nélkül szinte irreális lenne.

**A klasszikus módszer**

A kérdőív feldolgozásának első lépése az úgynevezett szociometriai mátrix összeállítása. Ez a szavazatok számlálásával végezhető el. Szokás a szavazatokat súlyozni aszerint, hogy

a válaszadó a felsorolásban hányadik helyen jelölte meg a kiválasztott személyt. A mátrix  $a_{i,j}$  eleme tehát azt adja meg, hogy a névsor  $i$ -edik egyede hány szimpátia-(antipátia-) pontot adott a  $j$ -ediknek. Ebben a táblázatban néhány egyszerű számításra van lehetőség. Ezekkel itt most nem foglalkozunk, csak ízelítőül megemlítjük a népszerűségi, kölcsönösségi, lejtési stb. indexeket.

A csoportstruktúra elemzésére használják a mátrix alapján megrajzolható gráfot, a szoci-

ogramot. Mivel a pontszámokat tartalmazó mátrix egy úgynevezett multigráfot (többszörös élek) határoz meg, melynek bonyolultsága – különösen nagy létszámmal, sok kérdésnél és kérdésenként sok válasznál – az elemzést megnehezíti, a szociogramot a következő úgy egyszerűsítéssel készítik el:

- A => Egyesítik a többszörös éleket
- B => Elhagyják az (irányított) élek közül azokat, melyeknek az ellentettje nem szerepel
- C => Az ellentétes élpárokat egyetlen irányítás nélküli élben egyesítik

Látható, hogy az így kapott egyszerű, irányítás nélküli gráf a kapcsolatokról kevesebb információt tartalmaz, mint az eredeti, hiszen csak a kölcsönös kapcsolatokat mutatja, súlyuk nélkül. Nagyobb létszámú csoportoknál még azt is megteszik, hogy csak bizonyos minimumot elérő pontszámokat vesznek figyelembe. E kényszerű szűrések nélkül azonban a manuális feldolgozás szinte kivihetetlen.

**A strukturális modell**

A szavazatokból konstruált teljes szociogram vizsgálatához néhány fogalomra van szükség, melyeket most vázlatosan ismertetünk kell.

A szociogram egy hírközlési vagy egy közlekedési hálózathoz hasonlóan két pont közötti közvetlen kapcsolatokat szemléltet. Lehetőség van azonban közvetett kapcsolatok létesítésére is. Így egy pontból más pontok érintésével távolabbra is eljuthatunk. Nem biztos azonban – különösen irányított gráfnál –, hogy az éleken a megjelölt irányban haladva minden pontból minden másikba eljuthatunk. Ha igen, akkor a gráfot (hálózatot) erősen összefüggőnek nevezzük.

A struktúra vizsgálatában fontos szerepük van azoknak a részgráfoknak – esetünkben a közösségen belüli kisebb csoportoknak, mikroközösségeknek –, amelyek egymással kölcsönös közvetlen vagy közvetett kapcsolatban vannak. A gráf egy ilyen tulajdonságú maximális részgrádját erősen összefüggő komponensnek, röviden komponensnek nevezik. A mikroközösségeket az jellemzi, hogy tagjaik egymással szorosabb kapcsolatban vannak, mint a közösség többi tagjával.

**Minta (Szavazólista)**

1. <b>Anci</b>	1.kérdés	Gizi	Mari	Vera
2. <b>Dóra</b>	1.kérdés	Flóra	Zsuzsi	Kati
3. <b>Erzsi</b>	1.kérdés	Kati	Dóra	Vera
4. <b>Flóra</b>	1.kérdés	Zsuzsi	Mari	Dóra
5. <b>Gizi</b>	1.kérdés	Anci	Vera	Mari
6. <b>Jutka</b>	1.kérdés	Piri	Dóra	Vera
7. <b>Kati</b>	1.kérdés	Anci	Vera	Mari
8. <b>Mari</b>	1.kérdés	Gizi	Anci	Vera
9. <b>Piri</b>	1.kérdés	Jutka	Dóra	Kati
10. <b>Saci</b>	1.kérdés	Jutka	Dóra	Kati
11. <b>Vera</b>	1.kérdés	Anci	Mari	Gizi
12. <b>Zsuzsi</b>	1.kérdés	Flóra	Dóra	Rozi

**Minta (Szavazatok)**

	0	1	2	3	4	5	6	7	8	9	0	1	2
1. Anci		-				3			2				1
2. Dóra			-	3				1					2
3. Erzsi			2	-				3					1
4. Flóra			1		-			2					3
5. Gizi		3				-		1					2
6. Jutka			2							3			1
7. Kati		3							-	1			2
8. Mari		2				3							1
9. Piri			2				3	1					-
10. Saci			2					3	1				-
11. Vera			3						2				-
12. Zsuzsi		1	2	3									-

**Minta (Kapcsolatok)**

	0	1	2	3	4	5	6	7	8	9	0	1	2
1. Anci		-			*				*				*
2. Dóra			-	*				*					*
3. Erzsi			*	-				*					*
4. Flóra			*		-			*					*
5. Gizi		*				-		*					*
6. Jutka			*							-		*	*
7. Kati		*								*			*
8. Mari		*				*							*
9. Piri			*				*	*					-
10. Saci			*				*	*					-
11. Vera			*			*			*				-
12. Zsuzsi		*	*	*									-





A legfontosabb formák, melyeket a strukturális modell alkalmazásával kimutathatunk, a következők:

1. Egyetlen elemből álló komponensek: a közösség periférikus egyedei, akik senkitől sem kapnak szavazatot.
2. Zárt komponensek: tagjai csak egymást választják, bár nem feltétlenül kölcsönösen.
3. Nyitott komponensek: tagjait preferáltnak egymást választják.
4. Klikkek: legalább három tagból álló komponensek, minden esetben kölcsönös választással.

A klasszikus modellben ettől eltérő, de nem ellentmondó kategóriákat használnak. Belátható ugyanis, hogy az ott megköltözött formáció (pár, lánc, csillag, háromszög) mindig egy komponensen belül található, bár a fenti mikroközösségeken belül több ilyen formáció is megjelenhet. Ezért a komponensek számítógépes szétválogatása megkönnyíti a klasszikus elemzést is.

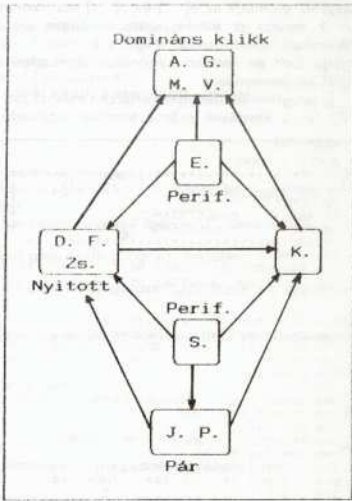
### Dominancia

A strukturális modell további előnye, hogy a gráf komponenseinek egymáshoz való viszo-

nyáról is szolgáltat információt. A komponensek definíciójából következik, hogy két komponens között — azaz egyedek között — nem lehet kétirányú kapcsolat. Esetünkben ez azt jelenti, hogy két mikroközösség között vagy nincs kapcsolat (izoláltak), vagy csak az egyiknek a tagjai szavaznak a másikéira (favorizálás). A komponenseknek ezt a viszonyát parciális rendezésnek nevezik. Itt a parciális jelző azt fejezi ki, hogy nem minden párnál dönthető el az alá-fölé rendeltség. Ezért a csoport mikroközösségei nem rangsorolhatók ugyan egyetlen sorozatba, de feltétlenül van legalább egy olyan komponens, melynek tagjai másokra nem szavaznak, de kapnak külső szavazatokat; ezek a maximális komponensek (domináns mikroközösségek). Ugyanígy léteznek minimális komponensek. Ilyenek többek között a periférikus egyedek.

A mikroközösségek viszonyainak, azaz a csoport makrostruktúrájának vizsgálatához olyan algoritmusokat kell alkalmazni, amelyekkel egyrészt a gráf komponensei szétválogathatók, másrészt a komponenseket úgy kell rendezni, hogy a dominancia megállapítható legyen. Mivel a parciális rendezettség nem ad egyértelmű sorrendet, ezért az úgynevezett kanonikus sorrendbe való rendezést kell megvalósítani:

### Minta



### Makrostruktúra

### Minta (Összefüggés)

	0	1	2	3	4	5	6	7	8	9	0	1	2
1. Anci		-				*			*				*
2. Dóra	*	*	-		*	*		*	*				*
3. Erzsi	*	*	*	-	*	*		*	*				*
4. Flóra	*	*	*		-	*		*	*				*
5. Gizi	*				-			*					*
6. Jutka	*	*	*		*	*	-	*	*	*			*
7. Kati	*				*		-	*					*
8. Mari	*				*		*		-				*
9. Piri	*	*	*	*	*	*	*	*	*	-			*
10. Saci	*	*	*	*	*	*	*	*	*	*	-		*
11. Vera	*				*		*	*				-	
12. Zsuzsi	*	*	*	*	*	*	*	*	*	*	*	*	-

- A => A komponensek tagjai egymás mellett legyenek
- B => Egyetlen komponenst sem veszünk sorra, amíg fel nem soroltuk mindazokat, amelyek a rendezési viszonylatban fölé vannak rendelve

### Számítógépes programcsomag

A feladat megoldásához célszerűen több kisebb programot mutatunk be. Ezért a programok egyszerűbben készíthetők el, s mivel áttekinthetőbbek, más gépre való átírásuk is könnyebben megvalósítható. A bemutatott változat a C Plus/4-en futó CBM BASIC V3.5 verziójában készült, amely a C64 Simon's BASIC-hez hasonló.

A programcsomag moduljai hat lemezes állományt hoznak létre, illetve használnak fel. Az állományok neve a csoportot azonosító, legfeljebb 12 karakteres névből és egy négykarakteres típusjelből áll. Ez utóbbiakat a program automatikusan generálja. Az állományok szerkezete:

- <csop>.NEV = n::=létszám / s\$[1..n]::=névsor
- <csop>.LIS = n::=létszám / k::=kérdések száma  
s\$[1..n,0..3k]::=szavazólista
- <csop>.SZA = n / s[1..n,0..n]::=szociometriai mátrix (A 0. oszlop a csoporton kívül)
- csop>.ADJ = n / a[1..n,0..n]::=adjacencia mátrix (A közvetlen kapcsolatok)
- <csop>.HAT = n / b[1..n,0..n]::=elérési mátrix (Az a[...]-edik hatványa)
- <csop>.ORD = n / c[1..n,0..n]::=a[...] kanonikus alak  
g[1..n,0]::=komponens indexek r[1..n]::=kanonikus sorrend

### Minta (Rendezett)

	0	1	2	3	4	5	6	7	8	9	0	1	2
1. Anci		-	●	●									
2. Gizi	●		-	●									
3. Mari	●	●		-	●								
4. Vera	●	●	●		-								
5. Kati	*		*	*		-							
6. Dóra					*		-	●	●				
7. Flóra			*				●	-	●				
8. Zsuzsi	*						●	●	-				
9. Erzsi			*	*	*					-			
10. Jutka			*	*							-	●	
11. Piri			*	*						●		-	
12. Saci			*	*						*		*	-



Az 1. modul a névsor és a szavazólista beírását és kimentését végzi. A 2. modul a szavazólistából előállítja az a[. . .] és b[. . .] mátrixokat. A 3. modul az elérési mátrixot állítja elő a Warshall-algoritmussal. Végül a 4. modul váltogatja szét és rendezi kanonikus sorrendbe a gráf komponenseit.

A program kérdésenként három választ tételez fel, a kérdések száma azonban szabadon

megadható. A 2. modul megkérdezi a figyelembe veendő szavazatminimimumot. Ha ezt 1-nek adjuk meg, akkor a szűrés elmarad.

A programcsomag kiegészítőit további kiszolgáló modulokkal, amelyek a szavazólistát, illetve a négy mátrixot kinyomtatják. Így készült a Minta nevű példához tartozó bemutató sorozat.

A program futási ideje a névsor és szavazó-

lista beírásán kívül mintegy öt perc. Ez természetesen a csoport létszámától és kisebb mértékben a kérdések számától is függ.

A programcsomaghoz készíthetünk olyan programot, amely a hagyományos kérdőívek kitöltése helyett párbeszédese formában gyűjti össze a szimpátiázavazatokat. A fentebb említett egyszerű mutatók, indexek a programcsomag által készített lemezes adatállományokból kiszámíthatók. Érdemes ilyen irányban is kibővíteni a csomag szolgáltatásait.

Dr. Hack Frigyes

```

0 REM 1. MODUL
10 REM *****
20 REM * NEVSOR + SZAVAZOLISTA *
30 REM *
40 REM * N:=LETSZAM
50 REM * K:=KERDESEK SZAMA
60 REM *****
100 SCLNCR
110 RS=CHR$(13) : NS=" " : L$=" " : LIS=""
120 PRINT "CSOPORT NEVE: ";
130 PRINT " (MAXIMUM 12 KAR)";
140 INPUT " "; CS$
150 IF CS$="" OR LEN(CS$)>12 THEN 140
160 PRINT " 1. NEVSOR BEIRAS
170 PRINT " 2. NEVSOR LEMEZROL
190 INPUT " ; T : ON T GOTO 300,500
200 GOTO 160
210 :
230 INPUT " CSOPORT LETSZAMA = "; N
240 IF N<2 THEN 300
320 DIM S$(N)
330 PRINT " NEVSOR BEIRASAI"
340 FOR J=1 TO N
350 : PRINT USING "###. "; J;
360 : INPUT S$ : IF S$="" THEN 350
370 : J=1
390 : IF J=1 OR S$(J-1) THEN 400
390 : S$(J)=S$(J-1) : J=J+1 : GOTO 390
410 : S$(J)=S$
420 REM --- NEVSOR LEMEZRE ---
430 OPEN "0,0,0,CS$+N$+ ".S,W" : PRINT#0,N
440 FOR J=1 TO N :PRINT#0,S$(J) :NEXT J
450 CLOSE 0 : GOTO 610
460 :
500 REM --- NEVSOR LEMEZROL ---
510 OPEN "7,0,7,CS$+N$+ ".S,R"
520 INPUT#7,N : DIM S$(N)
530 FOR I=1 TO N :INPUT#7,S$(I) :NEXT I
540 CLOSE 7
600 REM --- SZAVAZATOK BEIRASA ---
610 INPUT " KERDESEK SZAMA, = "; K
620 IF K<1 THEN 610
630 OPEN "0,0,0,CS$+L$+ ".S,W"
640 PRINT#0,N : PRINT#0,K
650 FOR I=1 TO N
660 : S$=S$(I)
670 : PRINT USING "#####. "; S$
680 : FOR J=1 TO K
690 : PRINT USING " ##. KERDES "; J;
700 : INPUT A$,B$,C$
710 : PRINT#0,S$R$;A$R$;B$R$;C$
720 : IF J=1 THEN S$=""
730 : NEXT J
740 NEXT I
750 CLOSE 0
760 END

```

1. modul

3. modul

```

0 REM 3. MODUL
10 REM *****
20 REM * ELERESI - MATRIX ( .HAT ) *
30 REM *
40 REM * FORRAS:KAPCSOLAT-M. ( .ADJ ) *
50 REM * N:=LETSZAM
60 REM *****
70 :
100 SCLNCR
110 INPUT "CSOPORT NEVE: "; CS$
120 IF CS$="" THEN 110
200 PRINT " 1. INPUT "CS$, "ADJ"
210 OPEN "7,0,7,CS$+ ".ADJ,S,R"
220 INPUT#7,N
230 DIM B$(N,N),CX(N,N)
240 FOR I=1 TO N : FOR J=0 TO N
250 : INPUT#7,CX(I,J)
260 NEXT J,I

```

```

0 REM 2. MODUL
10 REM *****
20 REM * SZAVAZAT-MATRIX ( .SZA ) *
30 REM * KAPCSOLAT MATRIX ( .ADJ ) *
40 REM *
50 REM * FORRAS:SZAVAZOLISTA ( .LIS ) *
60 REM * N:=LETSZAM
70 REM * K:=KERDESEK SZAMA
80 REM * S$(1..N,B$,S$K) LISTA
90 REM *****
100 SCLNCR : CLR
110 INPUT "CSOPORT NEVE : "; CS$
120 IF CS$="" THEN 110
130 OPEN "0,0,0,CS$+ ".LIS,S,R"
140 INPUT#0,N : INPUT#0,K : K$=K*K
150 PRINT " LETSZAM = "; N
160 PRINT " KERDESSZAM = "; K
170 PRINT " PONTSZAM MAX. = "; K3
180 INPUT " KAPCSOLAT MIN. = "; H
190 IF H<1 OR H>K3 THEN 180
200 PRINT " FELDOLGOZAS"
210 DIM S$(N,K3),AX(N,N)
220 PRINT " 1. INPUT "CS$, "LIS"
230 FOR I=1 TO N
240 : FOR J=0 TO K3
250 : INPUT#0,S$(I,J)
260 : NEXT J
270 NEXT I
280 CLOSE 0
300 PRINT " 2. SZAMLAS"
310 FOR I=1 TO N
320 : P=3
330 : FOR J=1 TO K3
340 : S$=S$(I,J)
350 : M=0
360 : FOR L=1 TO N
370 : IF S$=S$(L,0) THEN M=L
380 : NEXT L
390 : AX(I,M)=AX(I,M)+P
400 : P=P-1 : IF P=0 THEN P=3
410 : NEXT J
420 NEXT I
500 PRINT " 3. OUTPUT "CS$, "SZA"
510 PRINT " "CS$, "ADJ"
520 OPEN "7,0,7,CS$+ ".SZA,S,W"
530 OPEN "0,0,0,CS$+ ".ADJ,S,W"
540 PRINT#7,N : PRINT#0,N
550 FOR I=1 TO N
560 : FOR J=0 TO N
570 : PRINT#7,AX(I,J)
580 : PRINT#0,(AX(I,J)+H)
590 : NEXT J
600 NEXT I
610 CLOSE 7 : CLOSE 0
620 END

```

2. modul

```

270 CLOSE 7
300 PRINT " 2. WARSHALL ALG. "
310 FOR L=1 TO N
320 : FOR I=0 TO N : FOR J=0 TO N
330 : B$(I,J)=CX(I,J)
340 : NEXT J,I
350 : FOR I=0 TO N : FOR J=0 TO N
360 : LX=B$(I,L) AND B$(L,J)
370 : CX(I,J)=LX OR B$(I,J) OR I=J
380 : NEXT J,I
390 NEXT I
400 PRINT " 3. OUTPUT "CS$, "HAT"
410 OPEN "0,0,0,CS$+ ".HAT,S,W"
420 PRINT#0,N
430 FOR I=1 TO N : FOR J=0 TO N
440 : PRINT#0,CX(I,J)
450 NEXT J,I
460 CLOSE 0
470 END

```

4. modul

```

0 REM 4. MODUL
10 REM *****
20 REM * RENDEZETT KAPCSOL. ORD *
30 REM * G(N) CSOPORTINDEXEK
40 REM * R(I) ELERESI INDEXEK
50 REM *
60 REM * FORRAS:ELERESI M. ( .HAT ) *
70 REM * KAPCS. M. ( .ADJ ) *
80 REM * N:=LETSZAM
90 REM *****
100 SCLNCR
110 INPUT "CSOPORT NEVE: "; CS$
120 IF CS$="" THEN 110
130 PRINT " 1. INPUT "CS$, "HAT"
140 OPEN "7,0,7,CS$+ ".HAT,S,R"
150 INPUT#7,N
160 DIM CX(N,N),MX(N,N),G(N),R(N),F(N)
170 FOR I=1 TO N : FOR J=0 TO N
180 : INPUT#7,CX(I,J)
190 NEXT J,I
200 CLOSE 7
300 PRINT " 2. KOMPONENSEK"
310 K=1 : DO
320 : L=1
330 : IF G(L)=0 THEN 370
340 : L=L+1
350 : IF L<=N THEN 330
360 : GOTO 500
370 : G(L)=K
380 : FOR I=1 TO N
390 : IF I=L THEN 450
400 : E$=I : J=1
410 : DO WHILE J<=N AND E$
420 : E$=E$ AND CX(I,J)=CX(L,J)
430 : J=J+1 : LOOP
440 : IF E$ THEN G(I)=K
450 : NEXT I
460 K=K+1 : LOOP
500 PRINT " 3. KANONIKUS SORREND
510 FOR I=1 TO N : FOR J=1 TO N
520 : MX(I,J)=CX(I,J) AND (NOT CX(J,I))
530 NEXT J,I
540 RA=1
550 DO : I=0
560 : DO : I=I+1 : LOOP WHILE F(I)
570 : I=I+1
580 : IF MX(I,I) THEN 560
590 : J=J+1 : IF J<=N THEN 500
600 : F(I)=1 : R(RA)=I : RA=RA+1
610 : FOR J=1 TO N : MX(J,I)=0 : NEXT J
620 : I=0
630 : FOR L=1 TO N
640 : IF I=L OR G(L)<>I THEN 670
650 : F(L)=1 : R(RA)=L : RA=RA+1
660 : FOR J=1 TO N : MX(J,L)=0 : NEXT J
670 : NEXT L
680 LOOP WHILE RA<=N
690 REM 4. MODUL/FOLYTATAS
700 PRINT " 4. INPUT "CS$, "ADJ"
710 OPEN "7,0,7,CS$+ ".ADJ,S,R"
720 INPUT#7,N
730 FOR I=1 TO N : FOR J=0 TO N
740 : INPUT#7,CX(I,J)
750 NEXT J,I
760 CLOSE 7
800 PRINT " 5. OUTPUT "CS$, "ORD"
810 OPEN "0,0,0,CS$+ ".ORD,S,W"
820 PRINT#0,N
830 FOR I=1 TO N : IO=R(I)
840 FOR J=0 TO N : IO=R(J)
850 : CX=CX(10,IO)
860 : IF CX(IO,IO) THEN CX=2*CX
870 : PRINT#0,CX
880 NEXT J,I
890 FOR I=1 TO N :PRINT#0,G(I) :NEXT I
900 FOR I=1 TO N :PRINT#0,R(I) :NEXT I
910 CLOSE 0
920 END

```

# STARTOOL—EDASS — a legjobb?!

Ha igazán élni akarunk a C64-es jó tulajdonságaival, akkor át kell térnünk az assembly programozásra. Programozási munkánk megkönnyítésére különböző assemblereket használhatunk, mint a HELP+, az ASM, a PROFI—ASS 64, a HYPRA—ASS, a UNIASS vagy az EDASS. Ezek közül részletesen bemutatjuk azt az assembler fordítót, amelyiket a legjobbnak találjuk.

## STARTOOL—EDASS

SYBEX Verlag GmbH

Frank U. Müller

- EDASS. A betöltő program
- STARTOOL.TIT. A töltés közben a képernyőn látható kép
- STARTOOL.OBJ. A program. Önállóan is betölthető és futtatható
- INTERFACE.O. Induláskor betölti ezt a programot, ha megtalálható a lemezen.

A program és munkaterületei a \$8000—\$BFFF-ig helyezkednek el. Fordítás közben használja a kezettápuffert is. A program elindulása után megadhatjuk, hogy mekkora területet szeretnénk a BASIC program számára lefoglalni. Itt legalább 1000 bájtot adjunk meg. A BASIC-terület fölött kezdődik az EDASS tárolóterülete. Egyszer több forrásszöveg is lehet a memóriában. Ha a BASIC programunkból gépi kódú programrészeket szeretnénk használni, akkor ezeknek a programrészeknek a fejlesztéséhez az EDASS jól használható. Indításkor akkora területet foglaljunk le a BASIC programunknak, amekkorán a program és a változói is effernek. A forrásszöveg programot az EDASS a saját tárolóterületén helyezi el. A gépi kódú például \$C000-tól fordíthatjuk. Így lehetővé válik, hogy egyszerre legyen a memóriában a BASIC program, az EDASS, a forráskód és a gépi kód. Egyébként ezzel az assembler fordítóval lehet a memóriába, a lemezre és a lemezegység memóriájába is fordítani. Van REASSEMBLER utasítása, vagyis egy tetszőleges memóriaterületből forráskódú programot állíthatunk elő.

Az EDASS-ban a címkeket betűvel kell kezdeni. Használhatók

- decimális számok — 11247
- hexadecimális számok — \$FFFF
- oktális számok — @4613
- bináris számok — %10100101010101

A műveletek (+, -, \*, /, MOD, AND, OR, XOR) nem prioritási sorrendben, hanem balról jobbra haladva hajtódnak végre, de zárójeleket lehet használni.

## BASIC-BŐVÍTÉSEK

- | = £ — Szabad tárolóterület az EDASS-ban
- | > kifejezés — A kétbájtos kifejezés felső bájta
- | < kifejezés — A kétbájtos kifejezés alsó bájta
- | = [ . . ] — A memóriacím tartalma
- | = "String" — Az első karakter ASCII értéke
- | = kifejezés — A kifejezés decimális értéke
- | \$ = kifejezés — A kifejezés hexadecimális értéke
- | @ = kifejezés — A kifejezés oktális értéke
- | % = kifejezés — A kifejezés bináris értéke

## Utasítás

- lsave kezdőcím,végcím,"név" [d]
- Memória mentése a d=8—12 eszköze "név.O" néven. „d” nélkül a 8-as lemezegységre ment.
- A további utasításokban a „d” jelentése egy egységszám 8—12-ig. Ha elmarad, akkor az EDASS először a memóriában, utána a 8-as egységen keres.
- lsembler"név" [d] laS
- A megadott nevű forrásszöveget lefordítja, ha a forrásban nincs más-képp kijelölve, akkor \$C000-ra. lbE
- lbegin"név" lbE
- A tárban kijelöli a megadott nevű forrásszöveg helyét. lbY
- lbyte"név",kezdőcím,végcím
- A megadott területet táblázattá konvertálja. lcL
- lclear
- Az összes címke törlése. lcO
- lcold
- lhidegstart (SYS 64738).
- ldir [d]
- A lemez tartalomjegyzékének kiírása. ldI
- ldisk"parancs" [d]
- A parancs elküldése a megadott egységre
- lds [d]
- A lemezstátusz kiírása

- ldisplay
- A tárban lévő forrásszövegek adatainak kiírása. ldisP
- ledit"név" leD
- A megadott nevű forrásszöveg editálása.
- lerase"név" [,kezdősorszám] [,végsorszám] leR
- Az adott nevű forrásszövegből kitörli a megadott sorokat.
- leraseb leRb
- Blokk törlése.
- lexit leX vagy —
- Kilépés az editorból.
- lfind,"string" [,kezdősorszám,] [,végsorszám] lFI
- Az aktuális forrásszövegben megkeresi a megadott sztringet. NICHTS GEFUNDEN = Nem találtam.
- SUCHE FORTSETZEN? = Keressek tovább? lgen = "J", nem = "N".
- lgo (kif vagy címke)
- A megadott címen kezdődő gépi kódú program futtatása.
- lgod d, (kif, vagy címke)
- A megadott címen kezdődő gépi kódú program futtatása a d számú lemezegység memóriájában.
- linsert"név" [d] liN
- Az adott nevű forrásszöveget beilleszti az éppen editált forrásszövegbe az aktuális sortól
- linsertb liNb
- Blokk bemásolása az aktuális sortól.
- Ugrás a megadott sorra.
- llet (címke) = (kifejezés) liE
- Értékkadás címkeknek.
- llist"név" [d] liI
- d=3—7, a forrásszöveg kilistázása a megadott egységre.
- lload"név" [d] liO
- Az adott nevű forrásszöveg betöltése a memóriába.
- lmod p1,p2,[sorszám] lmO
- Az editor kijelzésének beállítása:
- p1 = 0 az aktuális sort jelző csik kikapcsolva
- p1 = 1 az aktuális sort jelző csik bekapcsolva
- p2 = 0 normál kiírású az aktuális sor
- p2 = 1 inverz kiírású az aktuális sor
- sorhossz = 40 vagy 80 karakter
- lnow lnE

- Az összes forrásszöveg törlése a memóriából. lreA
- lreass"név",kezdőcím,végcím
- A memória megadott területéből editálható forrásszöveget készít.
- lrename"régí név" = "új név" lRE
- Forrásszöveg átnevezése.
- lsave[@]"név" [d] lsA
- Forrásszöveg elmentése. Ha felülírást szeretnénk csinálni, akkor a lsave @ parancsot használjuk.

## EDITOR PARANCSONK

Az EDASS editora soreditor. A kurzor mindig a képernyő közepén lévő, jelzovonalakkal és/vagy inverz kiírással kiemelhető sorban van. A képernyő legfelső sorában található az éppen szerkesztett forrásszöveg neve, a kurzor sora. A forrásszöveg nem tartalmazza a sorszámokat, az editor automatikusan generálja azokat. A legfelső sorban láthatók az informatív és a hibáüzenetek.

A forrásszöveg egy sora négy mezőből áll:

- címkemezőből,
- mnemonik, illetve direktívamezőből,
- operandusmezőből,
- megjegyzésmezőből.

## A billentyűzet

- HOME — A kurzor a sor bal szélére kerül
- CLR — Törli a kurzor sorát
- CRSR billentyűk — Használatuk értelemszerű
- F1 — Beszúrás, javítás üzemmódváltás
- F2 — Beszúrás módban a kurzor sora alatti sort átmá-solhatjuk egy másik sorba
- F3—F4 — A kurzor a sor jobb-bal szélére áll
- F5—F6 — Le-fel lapozhatunk a forrásszövegben
- F7—F8 — Mezőléptetés jobbra-balra
- CTRL—A — Blokk kezdetének kijelölése
- CTRL—E — Blokk végének kijelölése
- CTRL—B — Blokkhatárok törlése
- CTRL—L — Címkező törlése
- CTRL—O — Operandusmező törlése

CTRL — K  
Balra nyíl  
↑ sorszám  
↑ A  
↑ E

— Megjegyzésmező törlése  
— I exit rövidítéssel  
— Ugrás a megadott sorra  
— Ugrás a forrásszöveg elejére  
— Ugrás a forrásszöveg végére

### A forrásszövegben elhelyezhető pszeudoparancsok, direktívák

\* = (kifejezés) — A fordítási cím beállításra  
\* + (kifejezés) — Megadott darabszámú bajt lefoglalása  
(címké) = (kifejezés) — Értékekadás címkének  
> (címké) = (kifejezés) — Értékekadás globális címkének  
.BYTE (kifejezés), ... — Bajtkod beépítése a tárgykódba  
(kifejezés), ...  
.DBYTE (kifejezés), ... — Két bajt beépítése HI, LO sorrendben  
(kifejezés), ...  
.DISK"parancs" [d] — Fordítások a parancs kiadódik a lemezegységnek  
— Feltétel nélkül befejeződik a fordítás  
— d = 3–7, a megadott egységre kiíródik a hiba  
.ERROR (d) — Név, E néven lemezegységre mentődnek a hiba-  
.ERROR"név" [d] — üzenetek  
— Visszakapcsol az eredeti hibakezelésre  
.ERROR O — A megadott nevű forrásszöveget befordítja a  
.FILE"név" [d] — tárgykódba

.IF (kifejezés)

.ENDIF — A .IF és .ENDIF közötti forrásszöveg csak akkor fordítódik, ha a kifejezés értéke nem nulla

.LBL [d] — Kiírja a címkék értékeit  
.LBSL [d] — Kiírja hexadecimálisan a címkék értékeit  
.LBL "név" [d] — Név, T néven kiment a címkék értékeit lemezegységre  
.LBSL "név" [d] — Név, T néven kiment a címkék értékeit lemezegységre  
.LIST (d) [sa] — D = 3–7. Kiírja a tárgykód hexabájtjait  
.LIST"név" [d] — Név, L néven kiment a tárgykód hexabájtjait  
.OBJ M — A tárgykódot a C64 memóriájába fordítja  
.OBJ D(d) — A tárgykódot a lemezegység memóriájába fordítja  
— d = 3–7, kiírja a tárgykódot a megadott eszköze  
.OBJ(d) [sa] — Név, O néven a lemezegységre fordítja a tárgykódot  
.OBJ"név" [d] — Név, O néven a lemezegységre fordítja a tárgykódot

.OBJ (memória), (periféria) — A fentiek kombinációja lehetséges, például .OBJ M, "teszt"

.PUSH"név" [d] — A címkék elmentése név, P néven  
.PULL"név" [d] — A címkék visszatöltése  
.TEXT "szöveg", (kifejezés), ... — Szöveg és/vagy kifejezés ASCII kódjának beépítése a tárgykódba  
.VIDEO "szöveg", (kifejezés) — Szöveg és/vagy kifejezés képernyőkódjának beépítése a tárgykódba  
.WORD (kifejezés), ... — A kifejezés beépítése a tárgykódba LO, HI sorrendben

### A listázás formázása

.FF — Lapemelés  
.FORMAT papírhossz, szöveghossz, üres sorok felül, bal margó, jobb margó — A listázás formája a nyomtatón  
.FORMAT O — A formázás kikapcsolása  
.PAGE kezdő oldalszám — Oldalszámzás bekapcsolása  
.PAGE O — Oldalszámzás kikapcsolása  
.PRINT "üzenet", (8 bites kifejezés) — Nyomatatáskor az üzenet íródik ki  
.TITLE (helyzet), "üzenet" — Nyomatatáskor az üzenet kiíródik minden lap tetejére  
.TITLE O — kikapcsolása

```
100 MIN=100 : MAX=0 : IF PEEK(49152)+PEEK(49153)<>289 THEN
  LDAD"F #C000.0" : B,1
105 PRINT"cr left vel" : POKE 53280,0 : POKE 53281,0
110 SYS 49152 : E=0 : N=N+1
120 FOR A=56583 TO 56588 STEP -1
130 E=E#256+PEEK(A)
140 NEXT
150 M=( (255#256+255)#256+255)#256+255
160 D=M-E
170 F=50/985248#D : F=INT(F#100)/100
180 IF F<MIN THEN MIN=F
190 IF F>MAX THEN MAX=F
200 PRINT"home crsr down FREKVENCIA = "F"crsr left "
210 PRINT"crsr down MIN = "MIN"crsr left "
220 PRINT"crsr down MAX = "MAX"crsr left "
230 FS=FS+F : FF=INT(FS/N#1000)/1000
240 PRINT"crsr down ATLAS = "FF"crsr left ",N
250 GOTO 110
```

### Hibázójelek

?ZURUECKSETZEN NICHT ERLAUBT — A fordítási cím kisebb az aktuálisnál  
?LABEL BEREITS DEFINIERT — Kétszer definiált címké  
?LABEL NICHT DEFINIERT — Nem definiált címké  
?UNERLAUBTES ARGUMENT — Érvénytelen argumentum. Kifejezés értéke nagyobb 256-nál vagy nagyobb 65535-nél  
?FEHLER IM BEFEHLSFORMAT — Hibás parancs  
?KEIN FREIER SPEICHER MEHR — Kevés a szabad memória  
?PROGRAMNAME FEHLT — Program név  
?MNEMONIK NICHT VORHANDEN — Nem létező mnemonik  
?UNBEKANNTRE ADRESSIERART — Ismeretlen címzés mód

### Használat közben ügyeljünk a következőkre:

— A nullás lapos címkéket az első hivatkozás előtt kell megadni  
— Nem célszerű az EDASS-ra vagy az általa használt területre fordítani programunkat.  
Futtatás előtt javasoljuk a forrásnyelvű program elmentését.  
Az EDASS lehetőségeit egy példaprogram segítségével mutatjuk be, amely a hálózat frekvenciáját méri. A mérés az teszi lehetővé, hogy a 24 órás valós idejű óra (TOD) a hálózat frekvenciáját használja. Mivel a hálózat frekvenciája nem pontos, ezért az óra késik. Referenciájának a processzor órajelét (985248Hz) használjuk fel. Lemérjük a CIA2 segítségével, hogy a TOD által mért egy másodperc alatt hány órajelciklus telik el, ebből kiszámolhatjuk a hálózat frekvenciáját.

Bakos Imre—Kelemen Róbert

```

* =          *C000
.OBJ        "F #C000"

:
NMI         = #FEF7
OLDNMI     = #FE47
CIA2       = #DD00
TOD10     = CIA2+8
TODM      = CIA2+9
TODP      = CIA2+10
TODD      = CIA2+11
ICR       = CIA2+13
CRA       = CIA2+14
CRB       = CIA2+15
TIMER_A   = CIA2+4
TIMER_B   = CIA2+6

:
SEI
LDA        #>NMI
LDY        #<NMI
STA        $319
STY        $31B

:
LDA        CRA
ORA        #900
STA        CRA ; 50 HZ

:
JSR        BEALL

:
VAR LDA    TOD10
VAR BEQ

:
JSR        BEALL

:
LDA        #FF
STA        TIMERB+1
STA        TIMERB
STA        TIMER_A+1
STA        TIMER_A

:
LDA        #210010001 ; TIMER A INDUL
STA        CRA
LDA        #201010001 ; TIMER B INDUL
STA        CRB

:
SECVAR LDX  TOD10
        LDA  TODM
        BEQ  SECVAR

:
LDA        #0
STA        CRA
STA        CRB

:
LDA        #>OLDNMI
LDY        #<OLDNMI
STA        $319
STY        $31B
CLI
RTS

:
BEALL LDA  CRB
        AND #57F
        STA CRB

:
LDX        #0
STX        TODD
STX        TODP
STX        TODR
STX        TOD10
RTS

:
.END

```

A kisbetűvel írt kurzorvezérlő utasítások a megfelelő billentyűk lenyomásával válthatók ki.

## 13. feladat: Számológép

Írjon programot, amely el-  
látja egy egyszerű zseb-  
számológép funkcióit (alaprü-  
veleteket és négyzetgyök-  
vontást végez)! Legyenek olyan  
billentyűk, amelyek a számó-  
lógép gombjainak felelnek  
meg, és ezek lenyomásakor a  
program viselkedjen a zseb-  
számológéppel azonos mó-  
don!

## Megoldás

Ennek a feladatnak a meg-  
oldása a sorozatban eddig  
közölteknél egyszerűbb. Az  
egyszerű számológépnek ál-  
talanban csak két számot kell  
tárolnia: a kijelzőn látható ér-  
teket és az azt megelőző érte-  
ket. Ugyancsak sokat segíthet  
az a felfedezés, hogy elegendő  
egy műveleti jel tárolása.

A számológép bekapcsolá-  
sa — a program indulása —  
után mindkét számot tároló

Sorozatunkat elsősorban középiskolásoknak szánjuk, de reméljük, hogy min-  
den olvasónknak tanulási lehetőséget és szórakozást nyújt.

A feladatok a Nemes Tihámé országos számítástechnikai verseny színvona-  
lának felelnek meg. Minden esetben olyat választunk, amely röviden, gyor-  
san megoldható, de a megoldáshoz ötletre van szükség. A megoldást mindig  
a következő számban közöljük.

Mivel változatosságra törekszünk, különböző programozási nyelveket hasz-  
nálunk. Az is előfordul, hogy egy feladatra több programnyelven is közlünk  
megoldást, ezzel is elősegítve az ismeretszerzést.

A szerkesztőség várja az olvasók, a versenyzők leveleit. A legutósebb  
program beküldését könyvtalvánnyal jutalmazzuk. Ne feledjenek azonban a  
programhoz leírást is mellékelni!

regiszter értéke nulla. A szá-  
mológép szokásos használá-  
takor először az első számot  
visszük be. Ezt egy műveleti

jel zárja le, majd kezdődik a  
második szám beírása. A mű-  
velet elvégzése és az ered-  
mény kijelzése a második  
számot lezáró egyenlőségjel  
leütésének hatására megy  
végebe. Ha a második számot  
nem egyenlőségjel, hanem  
valamilyen más műveleti jel  
zárja le, az megfelel a egyen-  
lőségjel és a műveleti jel egy-  
más utáni leütésének.

Ezeket a funkciókat a pro-  
gramban is hasonlóképpen va-  
lósítjuk meg. A program IBM  
PC számítógépre készült Turbo  
Pascal 4.0 nyelven, de fel-  
építése egyszerű, speciális  
utasításokat nem használ,  
ezért könnyen átirtható bár-  
mely más gépre.

A kijelzőn — a képernyő  
közepén megjelenő kijelzőte-  
rületen — mindig az „érték”

```

{-----}
{
{           Egyszerű számológép
{
{           Pintér Gábor
{
{           1988. 12. 16.
{-----}

program számológép;

uses
  Crt;

var
  { A kijelzőn látható érték: }
  érték      : real;
  { Az előző érték: }
  eérték     : real;
  { Műveleti jel: }
  { Értékel: '+', '-', '*', / }
  művjel     : char;
  { Új=true jelzi, hogy új szám
  beírása következik: }
  új         : boolean;
  { egész=true, ha a szám
  egészrészét olvassuk: }
  egész      : boolean;
  { tört a tört rész
  súlytényezőjét tartalmazza: }
  tört       : real;
  { Ebbe olvas be: }
  c          : char;

{ érték kijelzését végzi }
procedure kijelzes;
begin
  ClrScr; write(érték:30:10);
end;

begin (számológép)
  { Kijelző terület a képernyő közepére }
  TextMode(C040);
  ClrScr; Window(5,12,35,12);
  TextBackground(Red); ClrScr;

  { Kezdeti értékek }
  érték:=0; eérték:=0;
  művjel:='='; új:=false;
  egész:=true; kijelzes;
  repeat
  { Egy karakter beolvasása }
  c:=ReadKey;
  { Ha számot ütöttek le }
  if (c<'0') and (c<'9')
  then begin

```

```

{ Ha ez egy szám kezdete
  érték-et törölni kell }
  if új
  then begin
    érték:=ord(c)-ord('0');
    új:=false; egész:=true;
  end
  else
  { A szám egészrészét olvassa be }
  if egész
  then
    érték:=érték*10+ord(c)-ord('0')
  else begin
    { A szám tört részét olvassa be }
    érték:=érték
      + (ord(c)-ord('0'))*tört;
    tört:=tört/10;
  end;
  kijelzes;
end

else case c of
  'T','t': { Előjelváltás }
  begin
    { Ha ezt szám kezdeténél ütik le,
    törli a kijelzést }
    if új
    then begin
      érték:=0;
      új:=false;
    end
    else
    { Előjelváltás }
    érték:=-érték;
    kijelzes;
  end;
  '*','-','+','/','.' :
  begin
    { Az előző művelet elvégzése }
    case művjel of
      '+' : eérték:=érték;
      '-' : eérték:=eérték-érték;
      '*' : eérték:=eérték*érték;
      '/' : eérték:=eérték/érték;
    end;
    { A mostani műveleti jel tárolása }
    művjel:=c;
    eérték:=eérték;
    új:=true; { Új szám beírás következik }
    kijelzes;
  end;
  '.' : { Tizedesvessző }
  begin
    if új
    then begin
      { Ha ez a szám első karaktere,
      akkor törli a kijelzést }

```

```

    érték:=0; új:=false;
  end;
  { Az első tizedesjegy következik }
  tört:=0.1; egész:=false;
end;

'c','C': { Kijelző törlése }
begin
  érték:=0; új:=true; kijelzes;
  eérték:=0; művjel:='=';
end;

'e','E': { Utolsó érték törlése }
begin
  érték:=0; új:=true; kijelzes;
end;

's','S': { Négyzetgyökvonás }
begin
  érték:=sqrt(érték); új:=true;
  kijelzes;
end;

end (case);

{ A program végtelen ciklusban fut }
until false;
end.

```

újra változóba tárolt érték látható. A számításához szükségünk van még egy ugyanilyen változóra (értékre), amely az előző értéket tárolja, és a műveleti jelet tartalmazó „művel”-re.

A program törzse repeat-untill végtelen ciklusban működik. A ciklus elején beolvassunk egy karaktert. Ha ez számjegy, akkor az a kijelzőbe kerül, a szám egész részének első jegye lesz.

A program a többi értelmes karaktert case utasítással választja szét. Ha az valamilyen műveleti jel, akkor az előző és a mostani érték közötti művelet elvégzése következik. Ugyanitt értelmezzük az egyenlőségjelet is. A kijelzőn látható érték azonban még nem változik, de az „új” változó jelzi, hogy újabb érték beírása következhet.

Külön kell még vizsgálni a tizedesvessző kezelését. Ez az „egész” és a „tört” változók beállításával jelzi, hogy a következő leütött számjegy a szám első tizedese lesz. Az előjelváltás elvégzésénél csak arra kell ügyelni, hogy a szám kezdetén leütött előjelet helyesen kezeljük, ugyanis ekkor még a kijelzőn az előző érték látható.

A négyzetgyökvonás és az utolsó érték törlése egyoperandusos műveletekből áll, és nagyban hasonlítanak egymáshoz: mindkettő csak az „érték” változót módosítja.

A törlés gombnak csak az a funkciója, hogy a kezdeti értékeket visszaállítsa.

## 14. feladat: Nyolc királynő

Elhelyezhető-e egy sakk-táblán nyolc királynő úgy, hogy egyik se üsse semelyik másikat? Írjon programot, amely megadja az összes lehetséges elrendezést!

Itt szeretném olvasóink figyelmét felhívni a programbeírások fontosságára. Több olyan megoldás is érkezett, amelyből egyik vagy másik, esetleg mindkettő hiányzott. Ezek nem értékelhetők, még akkor sem, ha különben jól működő programokról van szó.

Pintér Gábor

# Nemes Tihamér

## Országos Középiskolai Tanulmányi Verseny



1. Szabó Dániel



2. Biczó Tibor



3. Ladányi József

Helyezés	Név/Iskola neve és címe	Tanár
1.	Szabó Dániel III. o. Árpád Gimnázium Budapest, Nagyszombat u. 19. 1034.	—
2.	Biczó Tibor IV. o. Zrínyi Miklós Gimnázium Zalaegerszeg, Rákóczi u. 30. 8900.	Horváth Attila
3.	Ladányi József III. o. Árpád Gimnázium Budapest, Nagyszombat u. 19. 1034.	—
4.	Dédesi Péter IV. o. Földes Ferenc Gimnázium Miskolc, Hősök tere 7. 3525.	Dusza Árpád

*Az első tíz helyezettnek, amennyiben felsőfokú oktatási intézménybe felvételizik és felvételi tárgya a matematika vagy a számítástechnika, e tárgyakból nem kell felvételi vizsgát tennie. Tekintve, hogy egy másodikos és egy első gimnazista is kiemelkedő eredményt ért el és őket az a verseny kiírása szerint nem illeti meg, különjutalmuk részvétel a Novosibirszki Fiatal Programozók Nyári Iskoláján.*

*Az alábbiakban közreadjuk az első forduló feladatait. Ezek megoldását és a második forduló (döntő) feladatait következő számunkban adjuk közre.*

1. Milyen számot ír ki a képernyőre a következő program? Indokold meg!

```

10 I=0
20 GOSUB 50
30 PRINT I
40 STOP
50 GOSUB 60
60 GOSUB 70
70 GOSUB 80
80 GOSUB 90
90 GOSUB 100
100 GOSUB 110
110 I=I+1
120 RETURN
    
```

Pontszám: 3 pont

2. Hol vannak a hibák az alábbi BASIC szubrutinban? Javítsd ki! Milyen állítást fogalmazatsz meg a változók tartalmáról és a T\$ vektor rendezettségéről a megjelölt helyeken (a helyes változatban)? A szubrutin a T\$(i) N elemű vektort rendezné növekvően. A DO WHILE ... LOOP ciklus magja mindaddig végrehajtható, amíg a WHILE mögé írt feltétel teljesül.

```

1000 REM BESZÚRÁSOS RENDEZÉS:
1010 FOR I=2 TO N
1020 ISS=T$(I): J=I-1
    
```

```

← 1.
1030 DO WHILE J>=1 OR ISS<T$(J)
1040 T$(J)=T$(J+1): J=J-1
    
```

```

← 2.
1050 LOOP
1060 T$(J)=ISS
    
```

```

← 3.
1070 NEXT I
1080 RETURN
    
```

Pontszám: 8 pont

3. Mit számít ki az alábbi program, ha az x változó (0 < x < 1) legfeljebb 2 tizedes törtjegyet tartalmaz? Milyen feltételiek teljesülése esetén fejeződik be a végrehajtás? Mit ír ki a program x=0,3 esetén?

```

i:=0; w:="0";
kiir("x="); beolvas(x);
r[0]:=kerakit(x*100);
repeat
i:=i+1;
    
```

5. **Oravec Róbert III. o.** Dusza Árpád  
Földes Ferenc Gimnázium  
Miskolc, Hőskő tere 7. 3525.
6. **Berendi Péter III. o.** Tóth László  
Fazekas Gimnázium  
Budapest, Horváth M. tér 8. 1082.
7. **Szentesi Áron IV. o.** —  
Árpád Gimnázium  
Budapest, Nagyszombat u. 19. 1034.
8. **Tornyi Lajos IV. o.** Tóth László  
Fazekas Gimnázium  
Budapest, Horváth M. tér 8. 1082.
9. **Istenes Péter IV. o.** —  
Árpád Gimnázium  
Budapest, Nagyszombat u. 19. 1034.
10. **Megyeri Gergely III. o.** —  
Árpád Gimnázium  
Budapest, Nagyszombat u. 19. 1034.
- Versenyen kívül indult és kiemelkedő eredményt ért el:  
6. **Maróti Miklós II. o.** Drevenka István  
Radnóti Miklós Gimnázium  
Szeged, Komócsin Z. tér 12. 6701.
- elvi 7. **Kémárki Máttyás I. o.** Dr. Sárkány Ernő  
Katona József Gimnázium  
Kecskemét, Dózsa Gy. u. 3. 6001.

```
r[j]:=r[j-1]*2;
if r[j]>=100 then begin r[j]:=r[j]-100; w:=w+'1' end
else w:=w+'0';
j:=0;
while ((r[j]<>0) and (r[j]<>r[i]) and (j<i)) do j:=j+1;
until not ((i<30) and (j=i));
kiirt('végeredmény:');w);
```

Pontszám: 6 pont

4. Mit csinál az alábbi algoritmus? Az algoritmus egy T\$(N,2)-es táblázat és a K1\$, K2\$ változókkal dolgozik. Az index függvény egy szóveg karakterei alapján 1 és N közötti egész számot generál. (Az 'amig' ciklus akkor áll le, ha az amig mögé irt feltélt már nem teljesül.)

```
Eljárás:
K:=index(K1$): KK:=0
Ha T$(K,1)<>' ' akkor
KK:=K
Ciklus
Ha K<N akkor K:=K+1
különben K:=1
amig KK<>K és T$(K,1)<>' '
Ciklus vége
Elágazás vége
Ha KK=K akkor Ki:"baj!!!"
különben T$(K,1):=K1$: T$(K,2):=K2$
Eljárás vége.
```

Pontszám: 6 pont

5. Az alábbi algoritmus egy utcai automata pénzvisszaadásának menétét írja le úgy, hogy a lehető legkevesebb számú érmét adja vissza. Az FT() vektor írja le, hogy az automata milyen érméket kezel, értéke (20, 10, 5, 2, 1).

```
Pénzvisszaadás (ÖSSZEG):
Ciklus I=1-től 5-ig
Ha ÖSSZEG>=FT(I) akkor DB:=INT(ÖSSZEG/FT(I))
Ki: DB;" db. ";FT(I);"Ft-os"
ÖSSZEG:=ÖSSZEG-DB*FT(I)
Elágazás vége
Ciklus vége
Eljárás vége.
```

Módosítsd a visszaadás algoritmusát, ha azt is tudjuk, hogy az automata-nak melyik pénzérméből hány darabja van, amit egy D() vektorban tárolunk!

Pontszám: 6 pont

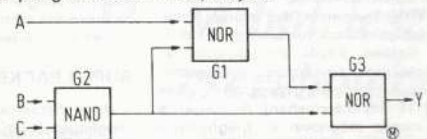
6. Mit rajzol az alábbi, LOGO nyelven írt program, ha teknőcünk kezdetben a képernyő közepén tartózkodik és felfelé néz? Mi a GVONAL eljárás szerepe? Mit tartalmaz a paramétere?

```
RIGHT 90 ABRA 30
az ABRA és a GVONAL eljárások LOGO definíciója:
TO ABRA :X
IF :X>9 THEN GVONAL :X LEFT 180 ABRA :X/2/3 LEFT 180 GVONAL :X
END
TO GVONAL :Y
REPEAT 180 [LEFT 1 FORWARD :Y*3.141592654/180]
END
```

Az utasítások jelentése:  
RIGHT fok, LEFT fok — jobbra, balra fordulás fok fokkal  
FORWARD LÉPÉS — elmozdulás, előre lépés hosszan  
REPEAT db [utasítások] — az utasítások megismétlése db-szer  
IF — a THEN-ág a sor végéig tart

Az utasításokat egymástól, illetve a paraméterüktől is szököz választja el. Az eljárások paramétereit elé :-ot kell tenni, az eljárásdefiníció kezdét a TO, végét az END alapszó jelzi.

- Pontszám: 7 pont
7. Egy elektronikus áramkörök gyártó vállalatnak digitális áramkörök hibás kapuit kell megkeresnie. Előzetes tapasztalatok alapján feltételezhető, hogy az ábrán látható áramkörben legfeljebb egy hiba van, és az úgy jelentezik, hogy valamelyik kapu kimenete a bemeneteitől függetlenül állandóan logikai 0-t ad ki (0-ba ragadt). Ennek megállapításához a G1, G2, G3 kapukból álló áramkör A, B, C bemeneteit a számított 63-as kimenetének 0., 1., 2. bitjére kötöttük, az áramkör Y kimenetét pedig a 64-es bemenet 0. bitjére:



Az egyes kapuk igazságtáblázata (NOR = nem vagy, NAND = nem és):

NOR		0	1	NAND		0	1
0	1	0	0	0	1	1	1
1	0	0	1	1	1	0	0

A következő programmal próbáljuk felfedni a hibákat:

```
10 REM HIBAKERESESE
20 FOR I=1 TO 3
30 READ X, Y, H$
40 OUT 63, X : REM X KIKÜLDÉSE A 63-AS KIMENETRE
50 IF (INP(64) AND 1) <> Y THEN 80 : REM A 64-ES BEMENETEN NEM Y JÖTT
60 NEXT I
70 H$="NINCS 0-BA RAGADT KIMENET"
80 PRINT H$
90 STOP
100 DATA ??, "G1 HIBAS"
110 DATA ??, "G2 HIBAS"
120 DATA ??, "G3 HIBAS"
```

Milyen számok kerüljenek a DATA sorokban levő kérdőjelek helyére? Válaszodat indokold meg!

- Pontszám: 8 pont
8. Egy liftet vezérlő program algoritmusát láthatod az alábbiakban. Gondold végig, hogyan működik ez a lift (ezt nem kell leírnod)! Sorolj fel minél több kritikus (kellemetlen) helyzetet, amelybe a lift utasa kerülhet az algoritmus miatt! A HOVA() vektor 32 elemű. (Hivni a liftet valamelyik emeletről lehet, parancsot adni pedig a fölkében.)

```
Liftvezérlő program:
SZINT:=0 : UTASÍTÁSSZÁM:=0
Ciklus
Várj amíg nincs HÍVÁS
Jegyezd föl(HÍVÓGOMB)
Ciklus amig UTASÍTÁSSZÁM>0
Menj oda(HOVA(UTASÍTÁSSZÁM))
Ajtó nyiss
Várj amíg eltelik 10 másodperc
Ajtó csukj
UTASÍTÁSSZÁM:=UTASÍTÁSSZÁM-1
Ciklus vége
Program vége.

Menj oda(X):
Ciklus amig SZINT<>X
Ha SZINT>X akkor
Menj le egyvel : SZINT:=SZINT-1
különben
Menj föl egyvel : SZINT:=SZINT+1
Elágazás vége
Ha van PARANCS akkor Jegyezd föl(PARANCSGOMB)
Ha van HÍVÁS akkor Jegyezd föl(HÍVÓGOMB)
Ciklus vége
Eljárás vége.

Jegyezd föl(GOMB):
Ciklus I=UTASÍTÁSSZÁM-tól 1-ig -1-esével
Hova(UTASÍTÁSSZÁM+1):=HOVA(UTASÍTÁSSZÁM)
Ciklus vége
UTASÍTÁSSZÁM:=UTASÍTÁSSZÁM+1
HOVA(1):=GOMB
Eljárás vége.
(A Várj amig... a Menj... az Ajtó... tevékenységek végrehajtására, a PARANCS-, ill. HÍVÓGOMB érzékelésére, valamint a PARANCS és HÍVÁS logikai értékek beállítására a lift közvetlenül képes, azokat nem kell definiálnunk.)
```

Pontszám: 8 pont (problémánként 1-1pont)

## Action Replay MK V

küli lemezekkel két perces másolás. A védtett lemezekkel kicsit tovább foglalkozik.

### FILECOPY

Fentebb már volt róla szó, maximum 249 blokkig másol fájlokat.

### FASTFORMAT

Nem túl kiemelkedő 20 másodperces idejével, hiszen intelligensebb másolóok ezt már 10 másodperc körüli idő alatt megteszik.

### BASIC TOOLKIT

Jelentősen bővíti a BASIC utasítás- és parancskészletet, jól alkalmazható parancsokkal (például DELETE, MERGE, APPEND, OLD, LINE), A funkcióbilleentyűket természetesen használja (LOAD, SAVE, DIR, RUN). Különböző listákat készíthetünk vele lemezről, egyenesen nyomtatóra és képernyőre egyaránt.

### TAPE TURBO

Kompatibilis több más kazettás turbóval, mint például az ABC turbóval és a Turbo Tape 64-gy stb. Érdekesége, hogy töltés közben nem kapcsolja ki a képernyőt.

### SPECIAL MONITOR

Olyan monitor, amely a program fagyasztása után alkalmazható, de működéskor nem változik a nulláslap és tarterület sem foglal.

### INTELLIGENTE HARDWARE

A kártya — funkciója ellenére — törökártya elleni védelmet tartalmaz, azaz ez a programokba beépíthető. Ezenkívül képes Centronics illesztőt kezelni a user porton keresztül.

A kártyához beszerezhető még egy lemez is, melynek programjai a fent felsorolt funkciókat támogatják (például DIASHOW, SPROTE EDITOR, BLOW UP, MESSAGE MAKER).

ta-csa-

Valószínűleg sokak által ismert — különösen crackerek körökben — az Action Replay törökártya-sorozat. Új tagja — lehet, hogy a megjelenésük már létezik újabb — az MK V-ös, minden eddigi elődjét felülmúlja a szolgáltatások sokféleségét és színvonalát illetően. Kezdő programozók és profik (crackerek) egyformán nagy hasznát veszik. Ára ugyan még Nyugaton is kissé borsos (119 DM márciusban), de megéri a pénzt, hiszen magnóval és meghajtóval egyaránt jól használható.

A doboz mögött a kártya lelke, egy LSI Custom chip, valamint 32 k ROM és 8 k RAM rejtőzik. Most pedig térjünk rá a funkciók bemutatására.

### RAMLOADER

A világ leggyorsabb sorozatban gyártott, soros átvitelű hajlékonylemez-gyorstöltője, még néhány párhuzamos átvitelűnél is gyorsabb. Lemezről huszonötöszer gyorsabban tölt, tehát mintegy kétszáz blokkot hat másodperc, kétszáznegyven blokkot két másodperc alatt, azaz nagy fájlokat is gyorsan tölt.

### TURBOLOAD

Lemenzen a másolást — betöltést, kimentést — szintén huszonötöszer sebességgel végzi.

### SPRITEKILLER

Kiváló lehetőség előre végignézni az arcade típusú játékok pályáit. Ha a játékot menet közben lefagyasztjuk (FREEZE), akkor a SPRITEKILLER-rel kikapcsolhatjuk a sprite-sprite ütközést.

### HARDCOPY

A program lefagyasztása után kinyomtatthatjuk a nagy felbontású (HIRES), illetve a multicolor képernyőt. A rutin a képet duplájára növelve, tizenhat árnyalattal jeleníti meg a nyomtatón. Sokféle nyomtatót kezel, például az MPS, EPSON, STAR család nyomtatói közül. Inverz kép is nyomtatható vele.

### PICTURE SAVE

Tetszőleges HIRES vagy MULTICOLOR képernyőt menthetünk lemezes fájlba. A rutin kompatibilis a Koala Painter, a Blazing Paddles, az Artist 64, az Image System rajz-programokkal.

### SPRITE MONITOR

A megtetszett sprite-okat a program fagyasztása után kitörölhetjük, kimenthetjük vagy átrakhatjuk más játékprogramokba. Érdekes lehetőség a sprite-animáció nyom követése.

### TRAINER POKES

A játék közben beírhatunk örökélet POKE-okat, energianövelő POKE-okat, sőt még meg is kerestethetjük ezeket a címeket, ahol az életeket, energiát számolja.

### MULTI STAGE TRANSFER

A rutinnal megnyitja az út a többrészes, kazettás programok bajlódás nélküli lemeze másolása előtt. A rutin ezt simán megteszi, és a kimentést már gyorsítva végzi el.

### SUPER PACKER

Nagy hatásfokú tömörítőrutin, ami kis tárigényével kitűnően alkalmazható száz blokknál nagyobb terjedelmű fájlok tömörítésére.

### TEXT MODIFY

A lefagyasztott programok szövegképernyőt szerkeszthetjük vele. Ezáltal lehetőség nyílik a programba épített utasítások magyarosítására, hi-score „módosítására”, akár maradandóan is.

### MONITOR

Az átlagos monitorutasításokon kívül van néhány speciális is: beépített konvertáló, bájtsorozat-keresés, összehasonlítás, eltérés, hexa dump. Akár futás közben is követhetjük, átirhatjuk a programokat. Érdekes lehetőséget kínál a programok különböző pontjaira elhelyezhető fagyasztási mutatók alkalmazása, melyekkel kielemezhetjük a programokat. A monitor ezeken kívül szintén széles körben támogatja a nyomtatókat a különböző listák készítésekor.

### DISKMONITOR

Magas színvonalú lemezmonitor, funkciói a szokásosak.

### DISKCOPY

A fent említett sebességgel védelem nél-

## Módosított másolás

Tapasztalatom, hogy a 80 k-ra bővített ZX-Spectrum gépeknél a legelterjedtebb 32 k-s lapozási megoldás a Rádiótechnika 1984. októberi számában közölt kapcsolással az OUT 191,0, illetve OUT 191,1. A legtöbb 80 k-s program ezt a lapozást használja. Gépetem öt éve bővítettük 16-ról 80 k-ra, ekkor a legkevesebb alkatrészt igénylő lapozási megoldás az OUT 253,N illetve OUT 125,N volt, ahol N=0...255.

Azoknak kívánok segítséget nyújtani a COPY LBL'80 k másolóprogram írásához, akiknek a gépük két különböző B/K címen lapoz. Ez a

másolóprogram mintegy 73,5 k befogadására alkalmas, kijelzése hex, kitöltéskor futó szám jelzi a még kitöltendő hossz.

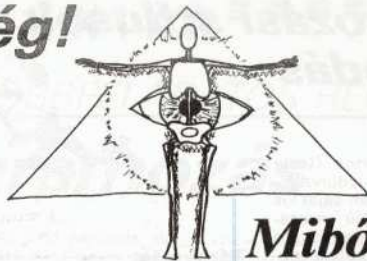
A lapozás átírásához 15 cím tartalmát kell megváltoztatni, miután megszüntettük egy erre alkalmas másolóval az automatikus indítást. Ismerve gépünk lapozási B/K címét, tekintsük A-nak azt, amelyik bekapcsoláskor gyakrabban jön le, és legyen B a másik 32 k-s lap. Ezután egyenként töltsük fel az alábbi címeket A, illetve B értékekkel, majd mentjük ki az eredeti névvel és automatikus indítással. Betöltés előtt célszerű mindkét lapot lehívni.

23840,A	24597,B	25256,A	26270,A
23885,B	24950,A	25577,B	26441,B
24097,B	25033,A	25639,B	26457,A
24104,A	25443,A	26069,A	

Kozma Ákos



# Nem ördögösség!



Sokszor hasznos lehet, ha eltüntetjük a kurzort úgy, hogy az irás azért látszik. Ezt megtehetjük egy utasítással: POKE 65286,91. Természetesen a kurzort vissza is hívhatjuk: POKE 65286,27.

Gyorsabbá tehetjük gépünk működését, ha leltjük a képernyőt:

POKE 65286, PEEK(65286) AND 239

és a visszaváltás:

POKE 65286, PEEK(65286) OR 16.

Egyébként ezt a módszert használja igen sok lemezmeghajtó-gyorsító is.

Ismét egy érdekes hanghatást mutat be a következő demo-program:

```
10 VOL8
20 FORI=1 TO 120:SOUND3,
  1023:1,2:NEXTI
30 FORI=993 TO 1023:SOUND3,1,2:
  NEXTI
```

Érdekes kipróbálni azt is, hogy a SOUND utasításban nem a 3-as, azaz a zajoscillátorra tesszük a hangot, hanem valamelyik zeneire (1-2).

Végül még egy érdekes program, amely a képernyőt mozgatja fel-le és jobbra-balra képpontként, azaz pixelenként. Érdemes a programot úgy továbbfejleszteni, hogy megoldja a függőleges finom szrotelt. Ennek feladatát az olvasókat bízom.

lme a program:  
10 FORI=1 TO 7  
20 POKE 65286, (PEEK(65286) AND 248) + I

30 FORG=1 TO 10:NEXTG:NEXTI

40 FORI=1 TO 7

50 POKE 65287, (PEEK(65287) AND 248) + I

60 FORG=1 TO 10:NEXTG:NEXTI

Elnézést kérek az előző cikkemben lévő hibáért, ezért most álljon itt egy helyesbítés, a program első sora helyesen:

1 VOL8

és a hanggal való kombinálás is helyesbítés:

61 SOUND1,0-100,1

Bácsi Péter

## Miből lesz a cserebogár?

Ezentúl szeretnénk minden számban egy kis válogatást közölni a legújabb, TOP-listára esélyes programokból.

Ezúttal négy, Amigára készült felhasználó programról szólunk néhány szót, melyek méltán vehetjük ki az Amiga-tulajdonosok elismerését.

### MODELER 3D

A MODELER 3D az Aegis Development legújabb nagy sikerű, grafikus programja. Segítségével háromdimenziós képeket készíthetünk, és ezeket az Aegis VideoScape 3D programmal animálhatjuk is. A program ezenkívül elfogadja a cég más grafikai programjaital készült grafikákat is, mint például az Aegis Draw, Aegis Draw Plus és Aegis Draw 2000 programok által készített grafikákat.

A program 1 Mb-át memóriát igényel, ezért ajánlott memóriabővítőt, hard drive-ot vagy másodmeghajtót használni.

### PRO SOUND DESIGNER

Új, professzionális zeneszerző eszköz az amigásoknak a Precision Inc. programja. A programmal professzionális dallamokat, effektusokat módosíthatunk, keverhetünk monóban és sztereóban egyaránt. A programcsomag magában foglalja a Midi-Plus programot, mellyel a felvett dallamokat visszajátszhatjuk egy ún. Midi-billentyűzeten.

### LATTICE C UPGRADE

Jó hír a C nyelv kedvelőinek! A Lattice Inc. piacra dobta az AmigaDOS C fordító 5.0-ás verzióját. A csomagban benne van az editor is. A bővített fordító teljes mértékben támogatja a 68020, 68030 processzorokat és a 68881 koprocesszort is.

### EXPRESS PAINT 3.0

Újabb segédlet grafikai mesterművek készítésére a Brown-Wagh Publishingtől. A program külön specialitása, hogy korlátlan mélységben törölhetjük az előzőleg elkészített grafikus elemeket. (Azaz egy kitűnő undo funkcióról van szó.) Korlátlan méretben és színezésben készíthetünk bonyolult grafikákat, mivel a program virtuális grafikus oldalkezelése lehetővé teszi, hogy ne fussunk ki a memóriából. Ezekben kívül alkalmas színkeverésre, nagytáras és teljes lappanoráma elkészítésére.

A BITKILLER a Commodore Magazin nyomán

## Egyből négyet C64

Programmal a kis felbontású grafika támogatható. Vagyis a grafikának az a fajtája, amikor egy karakterhelyet négy pontra bontunk. A program SYS 12 4096, mód,X,Y-nal hívható, ahol:

a mód: 0 pont törlése

1 pont rajzolása

2 pont "foggel-olása" (ha sötét volt világos, és viszont)

az X: x-koordináta (0-79)

az Y: y-koordináta (0-49)

A 0;0 koordinátájú pont a képernyő bal felső sarka.

A program működése:

220-270: paraméterek beolvasása és ellenőrzése

280-410: X,Y átszámítása karakterpozícióra, és a kurzor beállítása ide. Elkészül egyúttal a koordináták alapján a maszk is.

420-470: mi volt a képernyőn előzőleg? — Ezt kell majd módosítani.

480-610: a mód alapján a megfelelő módosítás elvégzése a maszk segítségével.

620-650: az új karakter kiírása és visszakeresés.

950-970: a karakterek képernyőkódjai sorrendben.

Földváry György

```

2 30: C000 .OPT P4
30: C000
90: C000
100: C000 CKCOM = #REFD
110: C000 GETBYT = #B79E
120: C000 CURSOR = #FFFB
130: C000 CURD1 = #01
140: C000 MD = #FF
150: C000 XK = #FE
160: C000 Y = #FD
170: C000 M1T = #FC
180: C000 M1RE = #FB
190: C000
200: C000 ** #C000
210: C000
220: C000 20 60 CB STWRT JSR BEOVL
230: C000 CA DEK
240: C000 06 FF STX MD
250: C000 20 60 CB JSR BEOVL
260: C000 30 0F BPL VEDE
270: C000 10 50 BPL VEDE
280: C000 06 FE JSR BEOVL
290: C000 14 30 54 BNI VEDE
300: C000 10 50 CPK #0B
310: C000 10 50 BPL VEDE
320: C000 14 30 54 TL Y
330: C000 0C 15 FE LUR X
340: C000 1E 20 01 RND #1
350: C000 0E 05 FC STA M1T
360: C000 3 E6 FC INC M1T
370: C000 04 10 FD LDA Y
380: C000 28 01 FC BEQ TOVA1
390: C000 0E FC AND M1T
400: C000 0E FC AND M1T
410: C000 0E FC AND M1T
420: C000 0E FC AND M1T
430: C000 0E FC AND M1T
440: C000 0E FC AND M1T
450: C000 0E FC AND M1T
460: C000 0E FC AND M1T
470: C000 0E FC AND M1T
480: C000 0E FC AND M1T
490: C000 0E FC AND M1T
500: C000 0E FC AND M1T
510: C000 0E FC AND M1T
520: C000 0E FC AND M1T
530: C000 0E FC AND M1T
540: C000 0E FC AND M1T
550: C000 0E FC AND M1T
560: C000 0E FC AND M1T
570: C000 0E FC AND M1T
580: C000 0E FC AND M1T
590: C000 0E FC AND M1T
600: C000 0E FC AND M1T
610: C000 0E FC AND M1T
620: C000 0E FC AND M1T
630: C000 0E FC AND M1T
640: C000 0E FC AND M1T
650: C000 0E FC AND M1T
660: C000 0E FC AND M1T
670: C000 0E FC AND M1T
680: C000 0E FC AND M1T
690: C000 0E FC AND M1T
700: C000 0E FC AND M1T
710: C000 0E FC AND M1T
720: C000 0E FC AND M1T
730: C000 0E FC AND M1T
740: C000 0E FC AND M1T
750: C000 0E FC AND M1T
760: C000 0E FC AND M1T
770: C000 0E FC AND M1T
780: C000 0E FC AND M1T
790: C000 0E FC AND M1T
800: C000 0E FC AND M1T
810: C000 0E FC AND M1T
820: C000 0E FC AND M1T
830: C000 0E FC AND M1T
840: C000 0E FC AND M1T
850: C000 0E FC AND M1T
860: C000 0E FC AND M1T
870: C000 0E FC AND M1T
880: C000 0E FC AND M1T
890: C000 0E FC AND M1T
900: C000 0E FC AND M1T
910: C000 0E FC AND M1T
920: C000 0E FC AND M1T
930: C000 0E FC AND M1T
940: C000 0E FC AND M1T
950: C000 0E FC AND M1T
960: C000 0E FC AND M1T
970: C000 0E FC AND M1T
980: C000 0E FC AND M1T
990: C000 0E FC AND M1T

```

```

630: C000 80 72 C0 LDA TMBL,X
640: C000 91 D1 STA (CURPO),Y
650: C000 68 68 VEDE RTS
660: C000 08B1 JSR CKCOM
670: C000 0060 JSR GETBYT
680: C000 0072 20 60 CB STWRT
690: C000 0072 20 60 CB STWRT
700: C000 0072 20 60 CB STWRT
710: C000 0072 20 60 CB STWRT
720: C000 0072 20 60 CB STWRT
730: C000 0072 20 60 CB STWRT
740: C000 0072 20 60 CB STWRT
750: C000 0072 20 60 CB STWRT
760: C000 0072 20 60 CB STWRT
770: C000 0072 20 60 CB STWRT
780: C000 0072 20 60 CB STWRT
790: C000 0072 20 60 CB STWRT
800: C000 0072 20 60 CB STWRT
810: C000 0072 20 60 CB STWRT
820: C000 0072 20 60 CB STWRT
830: C000 0072 20 60 CB STWRT
840: C000 0072 20 60 CB STWRT
850: C000 0072 20 60 CB STWRT
860: C000 0072 20 60 CB STWRT
870: C000 0072 20 60 CB STWRT
880: C000 0072 20 60 CB STWRT
890: C000 0072 20 60 CB STWRT
900: C000 0072 20 60 CB STWRT
910: C000 0072 20 60 CB STWRT
920: C000 0072 20 60 CB STWRT
930: C000 0072 20 60 CB STWRT
940: C000 0072 20 60 CB STWRT
950: C000 0072 20 60 CB STWRT
960: C000 0072 20 60 CB STWRT
970: C000 0072 20 60 CB STWRT
980: C000 0072 20 60 CB STWRT
990: C000 0072 20 60 CB STWRT

```

# Programozási stílusok különkiadás

## Előljáróban

Szerkesztőtársam, Bartos Gyula sorozatának sikerén felbuzdulva elhatároztam, hogy megírom saját kis egyrészes, hasonló témájú sorozatomat. S hogy több egybeesés ne nagyon forduljon elő, példaprogramjaimhoz a programnyelvek Trabantját, a BASIC-et hívom segítségül.

## Bevezetés:

Először is adva van egy számítógép és egy programozó. Kívülről-ként könnyű őket felismerni állandó nézetkülönbségükről. Az utasításokra a gép rendszerint visszadumál, többnyire angolul. A programozó viszont nem mindig ragaszkodik az angol nyelv használatához, de szerencsére a szóbeli részt a gép nem érti meg.

Ahány ember, ahányféle egyéniség, stílus, és a programozó is ember (ha nem is mindig látszik meg rajta).



A programozási nyelvek Trabantja a BASIC

## Tárgyalás:

Példaprogramom lényege, hogy a gép írassa ki az összes olyan valós számot, amelyek nagyobbak 1-nél és kisebbek 3-nál.

Aki nem hagyatkozik a számítógépre:

(1. lista)

Mások már kombinálnak is:

(2. lista)

A technikásabbak a számítógépre bizzák a döntést.

Közöttük is van struktúrákerülő, goto-mániás:

(3. lista)

Előfordulnak a modulrendszerrel kedvelő egyedek is:

(4. lista)

Mások szeretnek spórolni a helyllyel:

(5. lista)

S persze megtalálhatók a programozási tudásukat villogtatni kívánó egyedek is:

(6. lista)

## Befejezés:

S mindezek persze csak példák voltak izelítől a számítógépesek program nevezetű produktumainak irdatlan tömegéből. Befejezésként csak annyit:

Az itt fellelhető irományok kitalált személyek művei, s ha esetleg valaki magára ismer, az nem a véletlen műve. Direkt volt.

Sorozatomat ezzel zárom, s ígérhetem, hogy nem lesz folytatás.

SANY

### 1. lista

```
10 PRINT "2"
```

### 2. lista

```
10 FOR I=2 TO 2
12 PRINT I
14 NEXT I
```

### 3. lista

```
10 A=0
12 A=A+1
14 IF A>1 THEN GOTO 18
16 GOTO 12
18 IF A<3 THEN PRINT A
20 GOTO 12
22 GOTO 10
```

/csak a rágadás kedvéért/

### 4. lista

```
10 GOSUB 100
12 END
100 FOR I=0 TO 1000
```

```
102 GOSUB 200
104 NEXT I
106 RETURN
200 IF I>4 AND I<3 THEN GOSUB 300
202 RETURN
204 END
300 PRINT I
302 RETURN
304 END
```

### 5. lista

```
1 A=0
2 A=A+1:IF A>1 AND A<3 THEN PRINT A:GOTO 2:ELSE GOTO 2
```

### 6. lista

```
10 DIM A(10000) /szupercomputer/
12 FOR I=1 TO 10000:A(I)=I:NEXT I
14 FOR J=1 TO I
16 A(I)=INT(ABS(A(I)*100)/100)
18 IF A(I)>1 AND A(I)<3 AND A(I)=INT(ABS(A(I)))
THEN PRINT "A megoldás(ok):",A(I)=INT(A(I))/213.5*
(200+13+.5)
20 NEXT J
```



## TOP LISTA

játék programok	TOP LISTA							felhasználói programok		
	IBM	AMIGA	C-128	C-64	C-4(16)	SPECTR.	ENTERP.		TVC	APPLE
1.UCHlympics										
2.Grand Prix C										
3.Rocket B.										
4.Last Ninja 2.										
5.Ocean Ranger										
6.Robocop										
7.Techno Cop										
8.Zak McCrack										
9.Riziko /Risk/										
10.Bosszú										
1.Ceos 2.0										
2.Giga Paint										
3.Amiga Paint										
4.Art Studio.1.2										
5.Giga Cad										
6.Pagefox										
7.Rockmon. 5.5										
8.Printfox										
9.Windows										
10.Newsroom										

Listánkat felhasználói, illetve játéktékprogramokból állítjuk össze. A legjobbakat, legerdekesebbeket a beküldött javaslatok alapján rangsoroljuk. Ehhez kérjük az olvasók közreműködését. C64-re, ZX-Spectrumra, Enterprise-ra, TVC-re, Atarira és IBM-re készült programrangsorokat várunk havonta.

**Címünk:**  
Mikroszámítógép Magazin  
Szerkesztősége  
1371 Budapest, Pf. 433  
Diákszerkesztőség

# Konkurens programozás

A személyi számítógépek „személyiek” abban az értelemben, hogy egy felhasználó kiszolgálására és egyszerre csak egy feladat futtatására készültek fel. Ha több felhasználó egyidejű kiszolgálására nincs is szükség, mégis sokszor jó lenne, ha a nagyszámítógépeknél ismert többfeladatos környezet mikrogépeken is meg lenne. Bizonyos kis- és mikroszámítógépes operációs rendszerek (Unix, Xenix, MS-Windows, illetve újabban az OS2) már képesek egyszerre több feladattal is foglalkozni. E problémakörben elmélyedve fellapozunk tehát egy újabb fejezetet, folytatva mintegy ezzel A számítógépek motorja cikksorozatokat.

Néhány speciális számítógép-architektúra a működés gyorsítása érdekében eleve párhuzamos feladatvégrehajtásra készült fel. Ilyenek az ún. multiprocesszoros rendszerek, sejtprocesszorok, adatfolyamgépek stb. Sajnos ezek nagy része vagy még nem terjedt el igazán, vagy csak a fejlesztés stádiumában van.

Léteznek olyan programozási problématerületek — például diszkrét eseményszimuláció, folyamatirányítás, operációs rendszerek tervezése stb. —, amelyek megkövetelik, hogy egyes programrészeket futását külön feladatként lehessen megfogalmazni. A párhuzamos számítógép-architektúrák programozásához is feltétlenül szükségesek a megfelelő szoftvereszközök. A hagyományos magas szintű nyelvek nagy része erre nem ad lehetőséget. A speciális célú programnyelveken kívül az ALGOL68, a PL/1, az Ada, a Modula és a konkurens Pascal teszik lehetővé különálló, egymással párhuzamosan futó feladatok leírását, a konkurens programozást.

A következőkben ismertetjük az időosztás elvét, a párhuzamos feladatok közötti adatsere, kommunikáció és szinkronizáció elveit és módszereit, és leírunk egy olyan szubrutincsomagot, amely IBM PC-n lehetővé teszi C nyelven a konkurens programozást.

## Az időosztás elve

Először definiáljuk, mit értünk a továbbiakban feladaton! Ez a fogalom itt olyan különálló,

a számítógép erőforrásait (CPU, perifériák, memória) használó eljárást jelent, amely képes más feladatokkal logikailag párhuzamosan futni, velük kommunikálni és saját futását más feladatok futásával szinkronizálni. Valódi párhuzamos futás természetesen csak több végrehajtott egység — többprocesszoros — rendszerben lehetséges. A feladat általános fogalma egy logikai absztrakció, ahol az eljárás végrehajtásának időbeli lefolyásától eltekintünk. Az így definiált feladatok „szimulált párhuzamossága” már az egyprocesszoros rendszerekben is megvalósítható.

A nagyszámítógépek használatának elterjedésével a felhasználók kezdtek nehézkesebbek találni a korábban egyeduralkodó kötegelte adatfeldolgozást, különösen programfejlesztési célokra. Felmerült az igény a számítógép interaktív használatára, ami több felhasználónál csak időosztásos működtetéssel képzelhető el.

Időosztásnál egy felügyelőprogram váltogatja a feladatokat, egy feladattal a processzor csak egy bizonyos ideig foglalkozik. Feladatváltás akkor következik be, ha

— egy feladat olyan osztott használatú erőforrást (például mágnesszalagot) kíván használni, amely pillanatnyilag egy másik feladat számára van lefoglalva;

— egy feladat önként lemond a futásról és átadja a helyét egy másik feladatnak;

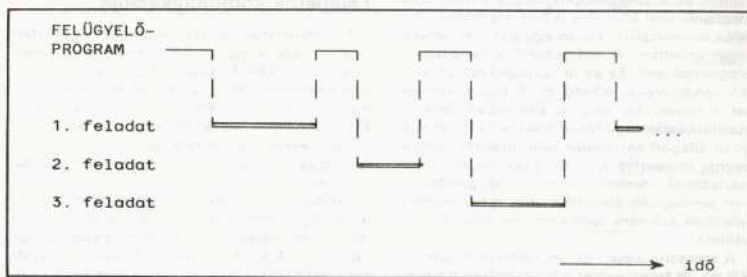
— lefejt a feladatra kiszabott időszeletet.

Az időosztás elvét az 1. ábra szemlélteti.

Nagyszámítógépeken a többfeladatos operációs rendszerek teszik lehetővé azt, hogy minden felhasználó úgy érezheti, mintha saját külön számítógépe lenne. Ezt a képzelte gépet a szakirodalom virtuális gépnek, az operációs rendszer virtuális gépeket kiszolgáló részét VM (Virtual Machine) operációs rendszernek nevezi. A valóságban erre a VM-rendszerre még további operációs rendszereket „ültetnek rá” (CMS, DOS OS/VS stb.), amelyek „ügy érzik”, mintha külön számítógépben, önállóan futnának (2. ábra). Igen hasznos lehetőség, hogy az egyes virtuális gépek egymásnak üzeneteket, adatállományokat is küldhetnek.

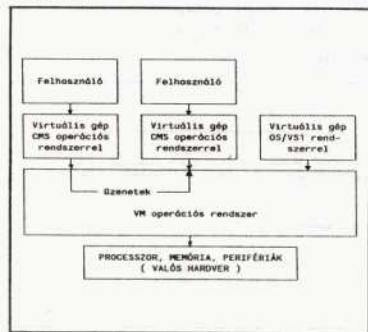
A mikroszámítógépeket egyszerre általában csak egy felhasználó használja, ezért nincs igény az erőforrások jobb kihasználására. Jó lenne persze egy-egy hosszabb fordítás vagy nyomtatás alatt szöveget szerkeszteni vagy akár valamelyik népszerű játékprogramot „tesztelni”, de ez az igény sokáig nem jutott el a gyártókhöz. Érdekes dolog, hogy míg az IBM PC AT processzora már tartalmaz többfeladatos programozás támogatására szolgáló elemeket, ezeket az általánosan elterjedt MS-DOS operációs rendszer nem használja ki. A Microsoft cég Windows rendszere viszont már egy „majdnem igazi” többfeladatos környezetet nyújt.

Sok helyen alkalmaznak lokális hálózatokat.

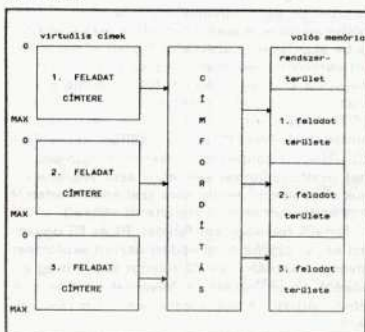


1. ábra

## 2. ábra



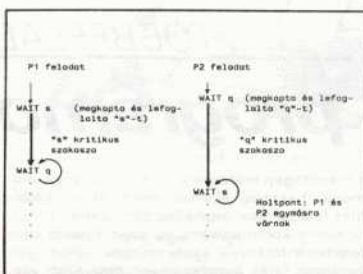
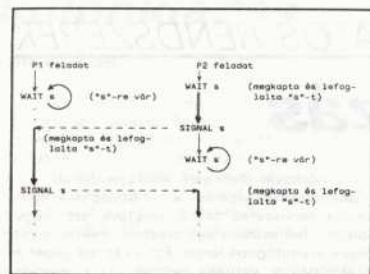
## 3. ábra



Ezek tartalmaznak egy kiszolgáló (server) gépet — rendszert egy nagyobb teljesítményű mikrogépet —, amely a hálózat többi gépétől érkező kérdéseket szolgálja ki. (Tipikus alkalmazás például, hogy a server gép valamilyen adatbázist tartalmaz, és a hálózat többi gépéről ezt le lehet kérdezni.) Ilyenkor elengedhetetlen, hogy a server gép képes legyen egyszerre több feladattal is foglalkozni.

## Többfeladatos rendszerek megvalósítása

Nem elegendő a CPU időosztását megvalósítani, szükséges az is, hogy az egyes feladatok „ne zavarják” egymást. (Képzeljük el például, hogy az egyik program felülírja a másik memóriaterületét, vagy közös kiviteli berendezés



4. ábra

használatakor az eredmény összekeveredik.) Minden egyes feladatnak saját különálló memóriaterületet kell hogy legyen, és egymás területét „nem is szabad látniuk”. Sok processzor — ilyen többek között az IBM PC at processzor, az Intel 80286 is — ún. címfordító rendszert tartalmaz. Itt a felhasználói programok címhivatkozásai virtuális címekként értendők, amelyek egy hardvermechanizmus „fordít le” valódi címekké. (3. ábra).

Megjegyzendő, hogy bizonyos nagyszámítógépes rendszerek, mint például az IBM 370-es sorozat gépei, a valós címter nagy részét 2 vagy 4 kb-ot laponként lemezen tartják. A felügyelőprogram feladata, hogy memóriahivatkozásnál a megfelelő területet „belapozza” a lemezről egy éppen nem használt memórialap helyére. Ez a tevékenység lassítja a rendszer működését, ezért a lapozás minimalizálása, a megfelelő lapozó algoritmus kiválasztása lényeges dolog, és fontos részét képezi az operációs rendszerek elméletének.

Szükséges továbbá, hogy bizonyos rendszerrutinok és a felügyelőprogram a címfordítási mechanizmust kikerülve is hozzáférhessenek a valós memóriához. Ezt és egy sor más lehetőséget azonban „el kell rejteni” a felhasználói programok elől. Ez az ún. privilegizált utasítások rendszerére érhető el. A processzorok két állapota van: normál állapotban csak az utasításokészlet egy része használható; privilegizált állapotban minden gépi utasítást végrehajthat (beleértve a címfordítást kikerülő memóriacímű utasításokat is). A felügyelőprogram privilegizált állapotban fut, a felhasználói feladatok számára csak a normál állapot használható.

A memória elején az ún. rendszertérlet található. Itt foglal helyet a felügyelőprogram, és itt vannak az operációs rendszer felhasználói programokból is hívható rutinjai (például B/K tevékenységek, memóriagazdálkodás, feladatvezérlés stb.). Ezeket nem gazdaságos minden egyes felhasználói területre bemásolni, elegendő, ha egy közös területen egy példányban megvannak. Ez a megoldás azonban további problémákat is felvet. Képzeliük el, hogy egy rendszerrutin futása közben történik feladatváltás, azaz a rutin hirtelen egy másik feladat számára kezd el dolgozni. Ha a rendszerrutin önmagát módosító kódot vagy olyan „belső” változót tartalmaz a közös adatterületen, amelyet az előző feladat kiszolgálása során módosított, megzavarhatja a következő feladat kiszolgálását. Ezért a rendszerrutinokat úgy kell megírni, hogy a tevékenységükhöz szükséges adatokat kívülről kapják, illetve futás előtt az aktuális feladat területéből „allokáljanak” memóriát belső változói számára. Az ilyen tulajdonságú programokat újrabélelhetőnek ne-

5. ábra

vezük. Sajnos az IBM PC-n általánosan elterjedt MS-DOS operációs rendszer rutinjai nem újrabélelhetőek, ezért MS-DOS alatt nehéz igazi többfeladatos környezetet létrehozni.

Nézünk meg, mi történik feladatváltáskor! A felügyelőprogram elmenti az állapotot (a regiszterek, jelzők és a programzámláló értéket), és a futásra várakozó feladatok közül kiválasztja a következőt. Beállítja az új feladat korábban elmentett állapotát és átadja a vezérlést neki. Utasítja egyben a címfordító mechanizmust, hogy a további virtuális címhivatkozások már az új feladat memóriaterületére vonatkoznak. Gondoskodni kell arról is, hogy magának a vezérlőprogramnak a futása ne legyen megszakítható. A feladatváltási tevékenységek egy része hasonló a megszakításkezeléshez (mint ahogy a megszakításkezelés is egy egyszerű többfeladatos rendszernek tekinthető): minden „rendes” megszakító rutin úgy kezdődik, hogy elmenti, és úgy végződik, hogy visszállítja a programállapotot (regisztereket).

## Feladatok kommunikációja

Többfeladatos rendszerekben az egyes feladatok csak a legritkább esetben működnek önállóan, a többi feladattól függetlenül. Általában szükséges, hogy az egyes feladatok egymásnak adatokat küldhessenek, illetve futásukat egymáshoz szinkronizálják. Ez a helyzet például akkor, ha két feladat ugyanazt az erőforrást akarja használni. Vizsgáljuk meg ennek részleteit!

Kritikus szakasznak azt a kódrészt nevezzük, amely egyszerre csak az egyik feladat számára dolgozhat, például egy erőforráskezelő programrészt. (A kritikus szakasz hasonló a vasúti sín azon szakaszához, amelyen egyszerre csak egy vonat tartózkodhat.) A kritikus szakaszokat az ún. szemaforokkal védjük. Ha a szemafor szabad, egy feladat beléphet a kritikus szakaszba, de ezzel egyidejűleg le is zárja — foglaltba állítja — a szemafort. Ha a másik feladat is be akar lépni, a szemafor foglalt állapota miatt várakozni kényserül. Ha az első feladat elhagyja a kritikus szakaszt, felszabadítja a szemafort. A szemafor tehát egy kétértékű változó, amelyen két művelet: a „vizsgáld meg és állítsd be” (WAIT) és az „állítsd szabadba” (SIGNAL) értelmezett. Ezeknek a műveleteknek oszthatatlannak kell lenni, azaz végrehajtásuk alatt semmi esetre sem szabad a feladattól a processzort elvenni (feladatot váltani).

Tegyük fel, hogy két feladat, P1 és P2 ugyanazt az „s” szemaforral védett osztott erőforrást kívánja használni, és P2 feladat kapja meg elsőként az erőforrást. A folyamat vázlatja a 4. ábrán látható. A dupla vonal a kritikus részt jelenti.

Szemaforok használatánál holtponthelyzet — dead lock — is kialakulhat, amikor két feladat egymásra vár. Képzeliük el, hogy P1 és P2 feladatok „s” „q” szemaforokat fordított sorrendben akarják allokálni. Ilyenkor a két folyamat elvileg végtelen ideig vár egymásra (5. ábra). A holtponthelyzet helyes útemező algoritmusmal elkerülhető, ha valamilyen erőforrás-elosztó program meg a feladatok elindítása előtt tudja, hogy azok milyen erőforrásokat kívánnak majd használni. Ez a helyzet a nagygépes OS operációs rendszerben, ahol ún. munkavezérlő — job control — utasításokkal meg kell adni a munka által használni kívánt erőforrásokat. Ilyenkor az OS vezérlőprogram gondoskodik párhuzamos munkák holtponthelyzetes futtatásáról.

Ezek után nézzük meg, hogyan lehet szemaforokat megvalósítani. Egy feladat háromféle állapotban lehet:

1. „Aktív” (futó) állapotban van a feladat, ha éppen fut a CPU-n. Aktív állapotú feladat egy processzoros rendszerben egyszerre csak egy lehet.

2. „Várakozó” állapotú a feladat, ha egy WAIT műveletet végzett egy foglalt állapotú szemaforon. A feladat mindaddig várakozó állapotú marad, amíg egy másik feladat SIGNAL műveletet nem végez el a szemaforon.

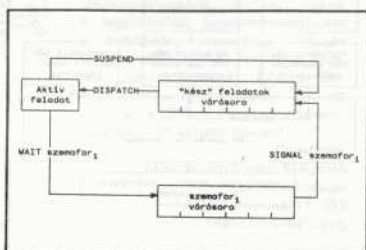
3. Egy feladat „kész” állapotú, ha logikailag készen lenne a futásra (semmilyen eseményre nem vár), de a processzor éppen egy másik feladattal van elfoglalva.

A futásra kész állapotú feladatok egy várakozó sorban várnak arra, hogy megkapják a vezérlést. Ha az aktív feladat futása valamilyen ok miatt megszakad — például egy WAIT művelet miatt várakozó állapotba kerül —, a felügyelőprogram útemező része ebből a várakozóból választ ki egy új feladatot a futásra (DISPATCH).

A kiválasztás stratégiája különböző lehet aszerint, hogy minden feladat azonos elbírálás alá esik, vagy bizonyos feladatokat magasabb prioritásúnak tekintünk. A magasabb prioritású feladatok gyorsabban választódnak ki, mint az alacsonyabb prioritásúak. Például a már említett VM operációs rendszer nagyon magas prioritásúnak tekint a B/K tevékenységeket kiszolgáló feladatokat, míg a nagy számítás igényű feladatok hátrányban vannak az interaktív felhasználók feladataival szemben. Természetesen sokkal egyszerűbb az első esetet megvalósítani egy FIFO (First In First Out, elsőként be — elsőként ki) szerkezetű listával, ami az érkező sorrendben való kiszolgálást teszi lehetővé. A cikk végén ismertetett konkurens vezérlőprogram is ezt a kiválasztási stratégiát követi.

A kész folyamatok során kívül minden szemaforhoz tartozik egy-egy várakozó sor azon feladatok számára, amelyek az adott szemaforra várakoznak. Ha egy szemaforon egy SIG-

6. ábra



NAL műveletet végez az aktív feladat, a váro- sor elején lévő feladat állapota „kész”-szé vá- lik, azaz elhagyja a sort, és a kész feladatok váro sorának végén foglalt helyet. Az aktuális feladat önként lemondhat a vezérlésről egy SUSPEND utasítással, ilyenkor a „kész”-sor vé- gére kerül, és egy új feladat kezd futni. A fel- adatok lehetséges állapotátmeneteit a 6. ábra mutatja. A DISPATCH művelet (új feladat kije- lölése futásra) automatikus, ha az aktív feladat elhagyja a processzort.

## A többfeladatos vezérlőprogram

A vezérlőprogram a többfeladatos operációs rendszerek legalsó rétege, amely szinkronizálja a feladatokat és kezeli a megszakításokat, vala- mint a B/K tevékenységet. Feladata hasonló az előzőekben említett VM operációs rendszeré- hez.

Az itt leírt mintarendszer komoly felhasználá- sra nem alkalmas, inkább csak demonstráció- sra céla készült. Nem használ időosztást, min- den feladatnak „önként” kell lemondania a ve- zérlésről. Ez bizonyos szempontból veszélyes lehet: ha egy feladat végtelen ciklusba kerül, az egész rendszer „beragadhat”. Előnye vi- szont, hogy megkerüli az MS-DOS operációs rendszer nem újrapelérhető voltból adódó problémákat. A rendszerben minden feladat egyenlő prioritással, a kiszolgálás érkező si- sorrendben (round-robin módon) megy végbe.

Nézzük meg, milyen adatszerkezetek szűk- ségek a vezérlőprogram számára! Először szükség van egy feladatleíró táblára (TCB, azaz Task Control Block), amely az egyes feladatok pillanatnyi állapotát írja le. Ez egy láncolt lista a következő elemekből:

- a feladat azonosítója (sorszáma),
- a feladat állapota (AKTÍV — VÁRAKOZÓ — KÉSZ),

- mutató a következő azonos állapotú fel- adatot leíró táblaelemre (ha van),
- mutató az előző azonos állapotú felada- tot leíró táblaelemre (kétirányú láncolás),
- mentési terület a regiszterek számára.

A vezérlőprogram Microsoft C nyelven ké- szült. Minden feladat egy paraméter nélküli C függvényként írható le. A C nyelv futatóren- dszere a függvények (szubrutinok) számára a rendszerveremből jelöl ki egy területet. Itt táro- lódnak a visszatérési címek és a függvény loká- lis változói. A veremterületre a BP regiszter mutat rá. Mivel az XT processzora, az Intel 8086 nem ad lehetőséget a virtuális memó- riakészítésre, nekünk kell az egyes feladatoknak azok létrehozásakor egy megfelelő nagysá- gú veremterületet kijelölni. A vezérlőprogram semmiféle védelmet nem tartalmaz erre a terü- letre nézve, így ha egy feladat túllépi a számá- ra megadott veremterületet, felülírja a köve- tkező feladat területét. „Komoly” vezérlőpro- gramoknál ez természetesen megengedhetlen lenne, szoftverből kellene gondoskodni a me- móriavédelemtől. Így azonban csak remé- nykedhetünk, hogy a feladat megelőzsi a szá- mára kijelölt veremterülettel (ez kisebb, de- monstratív jellegű feladatoknál megoldható). A módszert a 7. ábra szemlélteti.

A vezérlőprogram egy szubrutincsomag, amely az alábbi szubrutinokat (C függvénye- ket) tartalmazza:

- `init_tcb()` — a rendszer inicializálása
  - `create()` — létrehoz egy feladatot, lefo- lgalja a veremterületet és a TCB-t
  - `suspend()` — felfüggeszti az aktív feladat futását és kijelöli a következőt. Ha nincs több kijelölendő feladat, az eredeti tu- vább
  - `terminate()` — megszünteti a feladatot és felszabadítja a lefoglalt memóriaterületet
- A semafor is egy adatstruktúra, amelyet

SEMAPHORE\_TYPE-ként kell deklarálnunk. A semaforokat az alábbi szubrutinok kezelik:

- `<sem> = init_semaphore()` — iniciali- zál egy `<sem>` azonosítójú semafor
- `signal(<sem>)` — SIGNAL műveletet hajt végre az adott semaforon. A váro- sor első elemét a „kész” feladatok váro- sorának végére helyezi
- `wait(<sem>)` — WAIT műveletet hajt végre az adott semaforon. Ha a sema- for állapota szabad, semmi sem történik, ha foglalt, az aktív feladat belekerül a semafor váro sorának végére, és a „kész” feladatok sorának elejéről új fel- adat válik aktívá (ha van)

Megjegyzendő, hogy egy „igazi” vezérlő- program a felsoroltakkal jóval több funkcióra is képes (például időosztott várakozás, nagy prio- ritású várakozó állapot stb.).

Magának a vezérlőprogramnak a forrása túl nagyméretű ahhoz, hogy közölni lehessen, de egy alkalmazási példát itt láthatunk. Ebben ha- rom feladatot definiáltunk és indítottunk el. Az első karaktereket névbe a billentyűzetről, el- helyezi a „common” nevű változóban, majd jel- zést ad a kiíró feladatnak, nagybétű esetén a naplózó feladatnak. A kiíró feladat vár a jel- zésre, majd annak megérkezése után kiírja a „common” változót a képernyőre. Látható, hogy ebben az esetben a változó az osztott erőforrás, amelyet semaforral kell védeni. A harmadik (naplózó) feladat jelzések egy le- mezőjébe írja be a karaktert. Két semafor de- finiáltunk: a `p_log` a naplózó feladat jelzése, a `p_ask` a „common” változót védő semafor.

Szentjóni Ottó

## 2. lista

```

/*****
 * Ez a feladat írja ki a karaktert.
 *****/
p_task()
{
    printf("\nPRINTER FELADAT!\n");
    while(1)
    {
        /* Var a jelzere */
        wait(p_ask);
        /* If(common=="#") terminate();
         * else putchar(common);
         */
    }
}

/*****
 * Beolvassa feladat */
/*****
 * producer_task()
 *****/
{
    printf("\nBEOLVASO FELADAT!\n");
    /* "" karakterig fut. */
    while(common!="")
    {
        common = getch();
        /* Haja a kiíró feladatok */
        signal(p_ask);
        /* Ha nagybetu.jelz es log-nak */
        if(common=="A"## common=="Z")
            signal(p_log);
        /* Felfuggeszti magat */
        suspend();
    }
    return;
}

/*****
 * Log allomany keszito feladat */
/*****
 * log()
 *****/
{
    printf("\nLOG FELADAT!\n");
    /* Vegtelen ciklusban fut */
    while(1)
    {
        /* Var arra, hogy jelzessen neki */
        wait(p_log);
        while(common=="") {
            fgetc(stdin);
            terminate();
        }
        else fputc(common,op);
    }
}

```

## 1. lista

```

/*****
 *
 * Peldaprogram a kernelhez
 *
 * Harom feladatot indit el: a
 * beolvassa, a kiíro es a log-
 * allomanyt keszito feladatot.
 * Minden nagybetu karakter a log-
 * allomanyba kerül, amíg "-el"
 * le nem allitjuk a "log" felada-
 * tot. A kiíro feladat -#re ill le
 *
 *****/
#include <stdio.h>
#include <multi.h>

char common; /* Karakter helye */

/* Semaforok */
struct SEMAPHORE_TYPE p_ask;
struct SEMAPHORE_TYPE p_log;

/* LOG allomany */
FILE *op;

main()
{
    init_tcb(); /* Kernel init */

    /* Semaforok inicializolasa */
    p_ask = init_semaphore();
    p_log = init_semaphore();

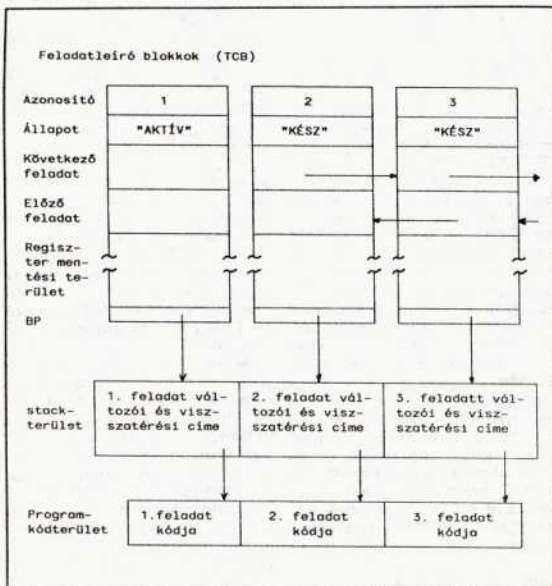
    op = fopen("log","w");

    /* Feladatok létrehozasa es elinditasa */

    create();
    p_task();
    create();
    create();
    log();
    producer_task();
    printf("\nVince tobb feladat!!!\n");
    exit();
}

```

## 7. ábra



# Rezidens óra

## Bájtok átalakítása

Az elkövetkezendő néhány alkalommal a bájtok átalakításával foglalkozunk úgy, hogy mindig egy gyakorlati példával be is mutatjuk az elméletet. A gépi kódú programozásnál ugyanis gyakran át kell alakítani a bájtokat mondjuk decimálisról hexadecimálisra, ASCII kódról képernyőkére, és néha éppen vissza. Köztudomású, hogy a C64 nem az ASCII kódot jeleníti meg a képernyőn, hanem úgynevezett képernyőkódot használ; ez a betűknél 64-gyel kisebb értékű az ASCII kódoknál, a számoknál pedig a képernyőkódok megegyeznek az ASCII kódokkal.

Erre valószínűleg azért szánta rá magát a Commodore cég, hogy az ASCII táblába elején (0-31) találhó kódokat, amelyek a nyomtatónak jelentenek vezérlőjeleket, a képernyőn viszont amúgy sem jeleníthetők meg, ne hagyja kárba veszni, hanem helyettük a betűket jeleníti meg. Ezért aztán ha a képernyő-memória területére akarunk betűket kiírni, akkor azok ASCII kódjából 64-et ki kell vonnunk. A művelet gépi kódnál rendkívül egyszerű, hiszen a 6. bitet kell csak törölnünk:

```
LDA ASCII—BETŰKÖD
AND #%10111111
STA KÉPKÖD
```

Ugyanígy, ha a betűk képernyőkódjából kívánunk ASCII kódot készíteni, akkor a képernyőkódot kell 64-gyel növelnünk:

```
LDA KÉPKÖD
ORA #%10000000
STA ASCII—BETŰKÖD
```

Ha számokat írunk ki a képernyő memóriaterületére (\$400-\$7E7, 1024-2023), vagy onnan olvasunk, akkor nem kell semmilyen műveletet végeznünk a KÉPKÖD-on, ugyanis az megegyezik a számok ASCII kódjával.

Az 1. listán látható a rezidens óra gépi kódú programja. Ebben a bájtok oda-vissza átalakítására volt szükség ahhoz,

hogy beállítható vagy leolvasható legyen a C64 órája (Time of day clock: TOD). Ez egy 24 órás időmérő, egytized másodperces felbontással. A TOD órájának bájtjai binárisan kódolt decimális (BCD) alakban jelennek meg a \$DD08-\$DD0B (56584-56587) memóriaterületen, a következő sorrendben: \$DD08 — 1/10 másodperc, \$DD09 — másodperc, \$DD0A — perc, \$DD0B — óra, ez utóbbinak a 7. bitje jelzi, hogy délelőtt vagy délután van-e. Az óra pontosságáért kezdőértéket írni csak bizonyos sorrendben célszerű: a TOD ugyanis leáll, amikor a \$DD0B tárolóba beírjuk az órák kezdőértéket, és mindaddig nem is indul el, amíg meg nem adjuk a tizedmásodperces kezdőértéket is. Emiatt utóljára az egytized másodperces kezdőértéket kell megadnunk. A TOD óráját egy közbelső tárolón (latchen) keresztül olvassuk le.

A közbelső tárolókba akkor másolja át a TOD a megfelelő memóriatárolóinak pillanatnyi értékét, amikor az órák értéket kiolvassuk a \$DD0B tárolóból, és mindaddig tart ez a közbelső tárolt (latchelt) állapot, amíg az 1/10 másodperceket is ki nem olvassuk a \$DD0B tárolóból. Ezekre figyelni kell a TOD órájának kezeléskor, különben előfordulhat, hogy az óra jár ugyan, de ezt nekünk nem mutatja, mert kiolvasáskor például elfelejtettük az egytized másodperces értékét kiolvasni.

Tekintsünk át, milyen átalakítások vannak a gépi kódú programban. Amikor az időpont kezdőértéket megadjuk, két-két számjegyből hoz létre egy-egy binárisan kódolt decimális, BCD számot (lásd az 1. lista 1620-1730-as sorait). Ennek a fordítottja történik az időpont megjelenítéskor: egy BCD számot alakít át kétjegyű decimális számmá és jeleníti meg a képernyőn (a lista 2020-2170-es sorai). Az időpont kezdőértéket

a „home” pozíciótól kezdődően a képernyőről olvassa le (\$0400-tól). Ezért a képernyő legfelső sorának legelső oszlopától kell az időpontot megadni: a következő formában: 19:35:41:05, s ez 19 óra 35 perc 41 másodperc 5 tizedmásodpercet jelent. Ha reggel 8 órát akarunk beállítani, akkor azt különféleképpen adhatjuk meg: 08:00:00:00 vagy 08:00:00 vagy 08:00:00 vagy 08:00:00 vagy 08:00:00. A szám végéről tehát a felesleges nullákat és a kettőspontokat is elhagyhatjuk. A 08 helyett azonban nem írhatunk 8-at, mivel az IDOKEZD alprogramnál két-két számjegyből állítja elő a program azokat a BCD számokat, amelyeket aztán a TOD órájának megfelelő rekeszeibe tesz a BEALLIT alprogramnál (lásd a lista 1560—1850, illetve 1000—1200-as sorait). Az időpont kezdőértéke alatt, új sorban, a SYS(49152) utasítással lehet az órák és a képernyőn a megjelenítést indítani.

Tételezzük fel, hogy a képernyő bal felső sarkába a következő időpontot írjuk be: 17:58:57, ami a következő képernyőkódnak felel meg: \$31, \$37, \$3A, \$35, \$38, \$3A, \$35, \$37. Az IDOKEZD alprogramban az IDOBE résznel (a lista 1620—1670-es sorai) először beolvassa a program az első szám képernyőkódját, a \$31-et, majd törli ennek a felső négy bitjét (marad \$01). Ezután átviszi az alsó négy bitet a felső négy bitbe — amivel előállította a BCD szám felső négy bitjét —, és ez most \$10-zel egyenlő. Ezután a képernyő következő oszlopából beolvassuk egy új számjegyét, például a \$37-et, s törli annak a felső négy bitjét — marad \$07 —, és ORA utasítással összeolvasztja a BCD szám felső négy bitjével: \$10-zel; \$17 lesz belőül, amit az IDOPONT+3 tárolóba tesz be, s már elő is állította az időpont óráit, a 17-et (a lista 1680—1730-as sorai).

Ha nincs kettőspont a képernyőn, akkor vége is az alprogramnak, mert a percek, másodpercek, tizedmásodpercek kezdőértéke 0-val lesz egyenlő. Esetünkben az alprogram átéli a kettőspont kódját, a \$3A-t (lásd a lista 1740—1770-es sorait), és a \$35, \$38-ból az előbbihez hasonlóan előállítja a \$58 BCD számot. Ez a perceket határozza meg, amit az IDOPONT+2 tárolóba helyez el. Hasonlóan állítja elő és teszi el a másodpercek értékét az IDOPONT+1-be, illetve a tizedmásodperceket az IDOPONT-ba.

Itt a BCD számon úgynevezett pakolt BCD számot értünk — hiszen a C64 ilyen számokkal dolgozik —, e számnak a felső és alsó négy bitje is egy-egy decimális számot ábrázol. Ennek értéke 0-tól 9-ig terjed. Így az egybájtos BCD szám maximális értéke \$99 lehet. A BCD számok fontos szerepet játszanak a mikroprocesszorok életében: külön utasításokkal — SED, CLD — hozzuk a processzor tudomására, hogy BCD számok kellene a műveletekhez.

Az óra időpontját a BEALLIT alprogramnál állítjuk be, ahol is a délelőtti, illetve a délutáni jelző 7. bitet is módosítani kell. A TOD órák — a korábban elmondottaknak megfelelően — a LEOLVAS alprogramnál olvassuk le, amit a képernyőre az IDOKIIR alprogram ír ki, decimális számokká alakítja vissza a TOD-ban megjelenő BCD számokat (a lista 1240—1430-as, illetve 1870—2300-as sorai). Ha a BCD szám felső négy bitje egyenlő nullával, akkor helyette a program egy helyközt — egy space-t — ír ki, a BCD alsó négy bitjét azonban mindig számként jeleníti meg. Az egytized másodperceket a program a képernyőre nem írja ki, mivel a program az IRQ ciklushoz kapcsolódik, és másodpercenként csak kétszer hajtódik végre.



# Programozási fogások és

# melléfogások



Cikksorozatnak ebben a részében nemcsak programozási, hanem egyéb melléfogásokról is szót ejtek. Többek között a sajátomról is. De kezdjük az elején...

A *Magazin* olvasóinak többségétől eltérően szívesen vállalkozom néhány tíz vagy száz programutasítást szolgáló bepytyögésére, ha az eredményben fantáziát sejtke. Eppen ezért kellett fel a figyelmemet a *Mikrovilág* idei 7. számában a *Form Writer* beharangozó írás. A RUN-ból átvett, valóban sokat ígérő program ismertetőjének elolvasása után — felretéve óvatosságomat, melyet a *Mikrovilág*ban között használhatatlan programlisták tömege alakított ki bennem, — amint gépközzebe juttattam, nekiláttam a "Lista" nevű program begépelésének. Ez a BASIC nyelven írt *Form Writer* működéséhez szükséges gépi kódú rutinokat olvassa be hexadecimális DATA sorokból, majd a szükséges átalakításokat elvégezve egy programfájlba írja őket. Az 1. listán látható programrészletet egy megjegyzés- és 47 DATA sor követi. Minden DATA sorban — az utolsó kettő kivételével — pontosan 60 karakter van, az utolsó előttiben 14, a legutolsóban az adatokat lezáró "—" található.

Gépiesen pytyögtem be a programot, csak a 70-es sor táján figyeltem fel arra, hogy valami nincs rendben. Ekkor számoltam meg az egyes DATA sorokból beolvasható karaktereket, és állapítottam meg, hogy a 15-östől az 50-esig terjedő programsorok feleslegesek. A következő sorokban levő karakterműveleteket sem értettem. Miért kell egyes karaktereket ilyen bizonyított módon kiszűrés a karakterláncokból, hiszen egyszerűen a DATA sorok szövegéből ki lehetett volna hagyni őket. A hexás kódok futólagos elemzéséből viszont azt derítettem ki, hogy a karakterek kihagyása teljes káoszhoz vezet.

Nem sajnáltam a fáradságot, megkerestem az eredeti programot. Az Amerikában kiadott *RUN* 1988. júniusi számában bukkantam rá. Ott a DATA sorokban tárolt karakterláncok — az utolsó két sor kivételével — 62 karakterből állnak, ugyanis minden huszadik hexadecimális számjegy után egy-egy

szóközzel van tagolva a lista. Ezeknek a szóközöknek a kiszűrésére szolgál a listán látható körülményes eljárás, amely a *Mikrovilág*ban közölt változatot használhatatlanná silányítja.

Az eredeti programot sem tudom fenntartás nélkül dicsérni. A két FOR ciklus csupán a ciklusváltozó végértékének formájában különbözik, mégis kétszer kell az olvasónak begépelnie. Ha a 15-től 50-ig terjedő sorokat elhagyjuk, a program működése nem változik, legfeljebb a végrehajtási ideje nő meg minimálisan, de ez elenyésző a belépésre fordított többletidőhöz képest. Ne hagyjuk figyelmen kívül, hogy a program végrehajtási sebessége itt lényegtelen, hiszen csak egyszer kell lefuttatni.

További egyszerűsítés lehetséges, ha az adatokat nem szóközzel, hanem vesszővel választjuk el. Ebben az esetben egy teljes DATA sor tartalma három READ utasítással olvasható be. Ekkor a program kódgeneráló része a 2. listán látható módon egyszerűsíthető. 19 sor helyett 9! Ezzel a módosítással a *Mikrovilág*ban közölt változat is hibátlanul működik.

Feltűnhet, hogy a DATA-ból beolvasott adatokhoz nincs ellenőrző összeg. Ez ebben az esetben elhagyható, mert a *RUN*-ban is, a *Mikrovilág*ban is a programok a BASIC ellenőrző kóddal lettek listázva, amely megbízhatóan működik, mint a DATA betöltőnél megszokott, egyszerű összeadáson alapuló ellenőrző összeg. A hibázás lehetőségét természetesen ez sem zárja ki.

Egyáltalán nem hiba, hogy az adatok végét a "—" karakterlánc jelöli, mégis érdemes megemlíteni, mert jól jellemzi a programozói gondolkodás útját. Itt bármilyen karakter állhatna, én például egy "\*" -ot használnék. A program írója gépiesen átvette a decimális adatokat feldolgozó betöltők szokásos záróértékét, anélkül, hogy gondolt volna arra, van-e ennek értelme. Ismétlem, hogy ez nem hiba, de jól alátámasztja jelen sorozatom mondanivalóját: *nem szabad elrejtett példákat követendőként az olvasók elé állítani*, mert rossz sablonok káros beidegződésekké válhatnak.

Háromféle melléfogással találkoztunk most. A programozóéval, aki fe-

leslegesen túlbonyolított egy egyszerű algoritmust, ezzel felesleges munkával terhelte olvasóit, a program adaptálójával, aki még a fáradságot sem vette, hogy a módosított programot tesztelje, és az eredményem, az olvasóéval, aki naivul azt hittem, hogy egy — „menő” folyóirattól átvett — jól használható programhoz jutok. Ezzel nem állok egyedül, másokat is megtévesztett a hangzatos cím: *Nem csak az IBM-é a világ!*

E cikk írásának napjaiban jelent meg a *Mikrovilág* idei 8. száma, a *Form Writer* BASIC programjának listájával. Az előző számban között programra vonatkozó helyreigazítással nem találkoztam benne. Valóban minden rendben van?

Barna László

```

0 REM CREATE EDITOR.64 ML FOR FORM W
  RITER
 5 OPEN B,B,B,"EDITOR.64 ML,P,W"
10 READ AS:IF AS="--" THEN CLOSE B:EN
  D
15 IF LEN(AS)>62 THEN 55
  BS=MIDS(AS,1,20)+MIDS(AS,22,20)+MI
  DS(AS,43,20)
25 FOR I=1 TO 30
  CS=MIDS(BS,(I*2)-1,2):HS=LEFT$(CS,
  1):LS=RIGHT$(CS,1)
35 H=VAL(HS):IF HS>"9" THEN H=ASC(HS)
  -55
40 L=VAL(LS):IF LS>"9" THEN L=ASC(LS)
  -55
45 BY=H*16+L:PRINT# B,CHR$(BY);
  NEXT :GOTO 10
50 IF LEN(AS)<21 THEN BS=AS:GOTO 70
60 IF LEN(AS)<42 THEN BS=LEFT$(AS,20)
  +RIGHT$(AS,(LEN(AS)-21)):GOTO 70
65 BS=LEFT$(AS,20)+MIDS(AS,22,20)+RIG
  HT$(AS,LEN(AS)-42)
70 FOR I=1 TO LEN(BS)/2
75 CS=MIDS(BS,(I*2)-1,2):HS=LEFT$(CS,
  1):LS=RIGHT$(CS,1)
80 H=VAL(HS):IF HS>"9" THEN H=ASC(HS)
  -55
85 L=VAL(LS):IF LS>"9" THEN L=ASC(LS)
  -55
90 BY=H*16+L:PRINT# B,CHR$(BY);
  NEXT :GOTO 10
  ...

```

## 1. lista

```

10 OPEN B,B,B,"EDITOR.64 ML,P,W"
20 READ AS
30 IF AS="--" THEN CLOSE B : END
40 FOR I=1 TO LEN(AS) STEP 2
50 HS=MIDS(AS,I,1) : LS=MIDS(AS,I+1,1)
60 H=ASC(HS)-48+7*(HS>"9")
70 L=ASC(LS)-48+7*(LS>"9")
80 PRINT#B,CHR$(16*H+L)
90 NEXT :GOTO 20
  ...

```

## 2. lista



## Hívja önmagát!

A rekurzív programozásról, feladatmegoldásról sokszor és sokan beszélnek számítógépes berkekben. A szokványos BASIC nyelvjárások alkalmatlanok vagy csak igen nagy megkötöttségekkel tesznek lehetővé olyan programozási fogásokat, amelyek legalábbis bemutatnák, illetve felhasználják ezt a lehetőséget. Az Enterprise IS BASIC-je — talán már nem is BASIC — támogatja ezt a programozási technikát.

Mit is jelent az eljárás? A számítás, illetve az algoritmus során a program a legegyszerűbb esetig — az eredeti — visszalép, egyszerűsíti a feladatot, és onnan kezd a megoldást. A kiindulási állapotig való visszalépéskor a részeredményeket vagy a feltételeket a számítógép tárolja. Minthogy esetenként a kiindulási feltételhez való visszalépéskor mindig ugyanannak a változónak adunk újabb és újabb értéket, és ezt az értéket csak egy bizonyos feltétel, meghívási mélységnél használja a program, ezért lokális változóként tároljuk. Az Enterprise képes erre (is)!

Nézzük meg közelebbről, hogyan? Az első programban a rekurzív működésének feltételeit, hatását próbáljuk vizsgálni (1. lista). Bemutatóként közöljük a programmal, hogy hányszor kívánjuk ugyanazt a blokkot, eljárás önmaga által alkalmazni, meghívni.

A program DEMO(N) eljárása a 170-es sor tanúsága szerint mindig egyet kisebb N érték felhasználásával hívja meg önmagát. A képernyőn leolvashatjuk az éppen aktuális N értéket. Folyamatosan követhető a veremutató értéke is. Látható, hogy egy DEF blokk önmagára való hivatkozása mintegy 1500 bájttal hosszabb területet foglal le a verem-

ben. Ebből következik, hogy egy eljárás a program hosszúságától függően maximum 38 alkalommal hívhatja meg önmagát. Ha enél is többször hívná meg magát az eljárás, akkor „összeakadna” a memória. A verembejegyzések felülírják a programot és az „el-szál”.

A 150-es sor kilépési — visszatérési — feltétele mindig csak az adott mélységből az egyvel korábbi szintre való visszatéréssel jár. Addig viszont az abban a mélységben érvényes — és csak ott érvényes — lokális értéket használja.

A rekurzív hívások alkalmazásának iskolapéldája a faktoriális kiszámítása. Ez definíció szerint

$$n! = n \cdot (n-1)!$$

Ha ezt az algoritmust követjük a faktoriális kiszámításához, akkor n! számításához előbb ismerni kell (n-1)! értékét, ahhoz viszont az (n-2)! értékét. Ne bonyolódjunk bele, bizzuk az egészet az Enterprise számítógépre. Mint a 2. lista programjának futási eredményéből láthatjuk, a számolásához nem is kezd addig, amíg vissza nem vezet a feladatmegoldást a legegyszerűbb, az 1! kiszámításához. Majd innen indulva sorba az adott meghívási mélység lokális változóival szoroz.

A rekurzíval megoldható feladatok másik iskolapéldája a bináris fa rajzolása. Erre programokat — Enterprise-ra is — közölt már a Magazin. Most nézzünk ismét egy példát. Rajzoljuk meg az ábra szerinti Koch-görbét. Igaz, ez a forma eltér a szokásos hópehely alaktól, noha megrajzolási algoritmus a attól alig különbözik. (Tudjuk jól, mit jelent a szá-

mitástechnikában az alig vagy a majdnem olyan hatással)

A kiindulási idom itt is egy egyenlő oldalú háromszög, de ennek oldalaira befelé rajzoljuk az oldal harmada hosszúságú újabb egyenlő oldalú háromszögeket, és mindig csak az újabban született vonalakat rajzoljuk meg. E munka algoritmus a görbe rendjétől függően egy adott mélységben a következő:

menj s/3 egységet előre  
fordulj 60 fokot jobbra  
menj s/3 egységet előre  
fordulj 120 fokot balra  
menj s/3 egységet előre  
fordulj 60 fokot jobbra  
menj s/3 egységet előre

Most már a Koch-görbe rendjétől függően csak (csak?) ismételni kell. A feladat rekurzíval történő megoldását a 3. lista mutatja.

Ennek a családnak a tagja a Hilbert-görbe is. Felfedezője azt bizonyította be, hogy a véges négyzet alakú síkdarab végtelen törtvonalal kitölthető úgy, hogy ez a vonal soha sem metszi önmagát és nem is záródik.

Minden magasabb sorszámmal jelzett rendű görbe — ezek egyre összetettebbek — mindig négy alacsonyabb rendű görbéből áll, de az n. rendű görbe és az n-1. rendű görbe körüljárási irányja ellentétes (ellenkező paritású). A 4. lista programja rekurzív támogatással biztosítja ennek a törtvonalnak a megrajzolását.

A kísérletező kedvű olvasó az ilyen, önmagát segítségül hívó — rekurzív — néhány soros programmal és az Enterprise-szal csodálatos dolgokat művelhet.

Sz. Lukács János

### 1. lista

```
100 ! rekurzio
110 INPUT PROMPT "hany futas ?":N
120 CALL DEMO(N)
130 END
140 DEF DEMO(N)
150 IF N=0 THEN EXIT DEF
160 PRINT N;256*PEEK(552)+PEEK(553)
170 CALL DEMO(N-1)
180 PRINT "dolgozz tovább"
190 PRINT N;256*PEEK(552)+PEEK(553)
200 END DEF
```

### 2. lista

```
100 ! faktoriális rekurzíval
110 INPUT PROMPT "N=" :N
120 IF N<1 OR N>35 OR N<>INT(N) THEN 110
130 LET S=1
140 CALL FAKTORIALIS(N)
150 END
160 DEF FAKTORIALIS(N)
170 IF N=0 THEN EXIT DEF
180 CALL FAKTORIALIS(N-1)
190 LET S=S*N
200 PRINT "(!;N;!)"=":S
210 END DEF
```

### 3. lista

```
100 ! kifordított KOCH görbe
110 GRAPHICS
120 OPTION ANGLE DEGREES
130 SET INK WHITE
140 PLOT 1000,201:PLOT ANGLE 180;
150 FOR I=1 TO 3
160 LET S=810:LET N=3 ! I<=n<=4
170 CALL KOCH(S,N)
180 PLOT RIGHT 120;
190 NEXT
200 END
210 DEF KOCH(S,N)
220 IF N=0 THEN PLOT FORWARD S:EXIT DEF
230 CALL KOCH(S/3,N-1)
240 PLOT RIGHT 60;:CALL KOCH(S/3,N-1)
250 PLOT LEFT 120;:CALL KOCH(S/3,N-1)
260 PLOT RIGHT 60;:CALL KOCH(S/3,N-1)
270 END DEF
```

### 4. lista

```
100 !HILBERT rekurzíval
110 GRAPHICS
120 OPTION ANGLE DEGREES
130 SET INK WHITE:LET P=1 !paritas
140 PLOT 940,700,ANGLE 180;
150 INPUT PROMPT "a görbe rendje 1<n<6":N
160 LET S=10*2^(6-N) !oldalhossz
170 CALL HILBERT(S,N,P)
180 END
190 DEF HILBERT(S,N,P)
200 IF N=0 THEN EXIT DEF
210 PLOT LEFT 90*P;
220 CALL HILBERT(S,N-1,-P)
230 PLOT FORWARD S:RIGHT 90*P;
240 CALL HILBERT(S,N-1,P)
250 PLOT FORWARD S: !osszekotes
260 CALL HILBERT(S,N-1,P)
270 PLOT RIGHT 90*P:FORWARD S;
280 CALL HILBERT(S,N-1,-P)
290 PLOT LEFT 90*P;
300 END DEF
```

## Kalandozások a karaktermemóriában

Ahogy a láthatatlanság elbűvöli fantáziánkat, úgy valamit láthatóvá tenni, szintén izgalmas. Különösen a számítógépen. Tegyük hát „láthatóvá” az E-128 karakter RAM-ját, hogy érthetőbbé váljon a karaktermemória felépítése.

Az E-128-nak alaphelyzetben 128-féle megjeleníthető karaktere (ASCII kód 32-159) van. Az 1-31 kódok a különböző vezérlő, azaz meg nem jeleníthető karaktereknek vannak fenntartva.

Hogyan alakul ki a képernyőn egy karakter képe? A karakterek egy 9x8-as ponthálozatban jelenítődnek meg, 9 bájtt 8 bitje segítségével. Azok a bitek, melyeknek értéke 1, tinta-, a 0 értékűek pedig papírszínűek lesznek.

A 128 darabos karakterkészlet képét ekképpen meghatározó bájtok a karakter ROM-ban vannak, ahonnan a számítógép bekapcsolásakor az operációs rendszer átmozdítja a RAM-ba. Így az E-128 eleve lehetővé teszi a karakterátdefiniálást a felhasználó számára, ellentétben például a C64-gyel. Egy-egy karakter megjelenítésekor az operációs rendszer a karakter RAM-ból átmozdítja a megfelelő 9 bájtot a képernyő-memória megfelelő helyére.

A karakter RAM az alapkiépítettségű E-128-ban a 46208-as címen kezdődik, és értelemszerűen 128x9 bájttal rendelkezik.

A karakterek képét hordozó bájtok elhelyezkedését a RAM-ban az 1. ábra mutatja. A jelölések, például az „ASCII 66/3” azt jelentik, hogy a 46530-as memóriacímen a „B” karakter reprezentáló kilenc bájtból a harmadikat találjuk.

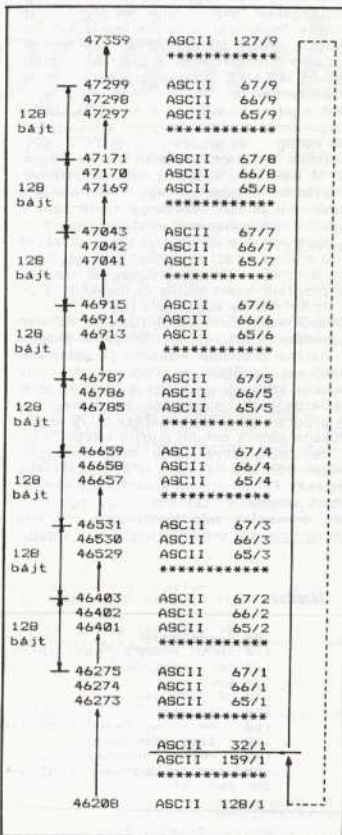
A karakter RAM tartalmát a *listán* szereplő programmal „kinagyítva” meg is jeleníthetjük, miáltal láthatóvá válik, hogy az egyes karakterek valójában hogyan is helyezkednek el a 9x8-as ponthálozatban. Ez hasznos lehet egyrészt a karakterek átdefiniálásánál, az új karakterek képeinek megtervezésekor, másrészt az egyéni érdeklődőknek is.

A program a karakter kinagyított képén kívül — amiből kettőt mutatoba a 2. ábrán láthatunk — kiírja a karakternek a memóriában elfoglalt sorszámát, az ASCII kódját (az utóbbit két adat az 1. ábrából következően 32-127-ig megegyezik), továbbá magát a karaktert, szokásos méretben, valamint a kinagyított karakterkészletet tároló memória címét és annak tartalmát.

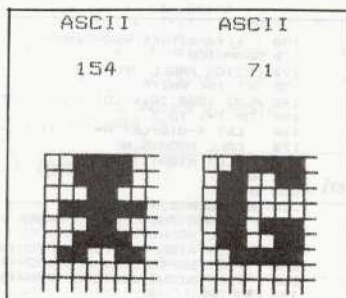
Losonczy János

```

120 TEXT
130 LET C$="A karakternek a memóriában
elfoglalt sorszáma"
140 SET CHARACTER
143,255,128,128,128,128,128,128,0;!
A nagyított karakter képen a 143-as kódú
karakter fogja mutatni a papír-, a
142-es pedig a tinta színt
160 PRINT C$;" ?";" (0-127)":INPUT
PROMPT "":SOR
170 TEXT
180 LET BASE=46208+SOR;! A karakter RAM
vizsgálatának kezdete
200 LET DB=128:;! 128-as karakter készlet
220 FOR H=BASE TO BASE+DB-SOR-1
225 ! ***** A karakter RAM karakterei
260 LET B=H-BASE+SOR
270 PRINT C$;"=";B
280 IF B<32 THEN LET B=B+DB
300 PRINT "ASCII kód=";B
320 PRINT "Karakter=";CHR$(B)
380 PRINT
400 FOR I=H TO H+B*DB STEP DB
420 ! ***** 1 karakter 9 bájttal
440 LET A=PEEK(I):LET B$=""
480 FOR J=7 TO 0 STEP -1
500 ! ***** 1 bájttal 8 bitje
510 LET A$=CHR$(143)
520 IF A-2^J>=0 THEN
540 LET A=A-2^J:LET A$=CHR$(142)
560 END IF
580 LET B$=B$&A$
600 NEXT J
620 PRINT B$;I;PEEK(I)
640 NEXT I
660 PRINT:PRINT:PRINT
680 NEXT H
700 END
    
```



1. ábra



2. ábra

## ENTERPRISE-TOTÓ 1989

A CENTRUM Áruházak Vállalat a Mikroszámítógép Magazinval közösen kétfordulós játékpályázatot hirdet. A kérdésekre a TOTÓ ismert szabályai szerint 1, 2, illetve X jelöléssel kell válaszolni. Beküldendő a 14 helyes válasz a kérdések sorrendjében a következő címre:

**CENTRUMNAGYKER**  
1431 Budapest, Pf. 195.

Minden megfejtést értékelünk. A játékosok mindkét fordulóban külön-külön nyerhetnek.

A találatlások szelvények kitöltői között a CENTRUM fordulónként az alábbi nyereményeket sorolja ki:

**1 db Spectrum-emulátort,**  
**3 db Enterprise-egert**  
**(mouse),**  
**10 db új programcsomagot**  
**(kazetta).**

Aki mind a két szelvényt helyesen töltötte ki, az a szupersorsoláson is részt vesz, és az őszi BNV-n **EPSON RX 80 típusú nyomtatót** nyerhet. Vigaszdíj is lesz.

Az első TOTÓ beküldési határideje 1989. július 25. Ne felejtse a megoldással együtt beküldeni nevét és pontos címét. A vigaszdíj sorsolásán a találatoktól függetlenül az vesz részt, aki gépének gyártási számát is feltünteti.

**KÉT FORDULÓ,**  
**HÁROM SORSOLÁSI!**

**ENTERPRISE**  
**COMPUTERS**

GMBH

1. Hány pixel a NICK video processzor maximális képfelbontó képessége interlace alatt?

1 672 x 256       2 672 x 512       3 640 x 512     

2. Hány oktávós a DAVE sound chip stereo hangja?

1 8       2 4       3 6     

3. Hány kbájtos a Local Area Network "NET" beégetett hálózati szoftver?

1 8       2 16       3 12     

4. Milyen elven működik az Enterprise számítógép tasztatúrája?

1 hall-generátor       2 optó-csatolás       3 fólia-tasztatúra     

5. Hány MHz a standard Enterprise számítógépben működő processzor sebessége?

1 6       2 4       3 8     

6. Hányadik oldalra helyezi el a program a rendszerváltozókat?

1 254       2 248       3 255     

7. Mi a neve a Hollandiában megjelenő Enterprise Magazinnak?

1 Enter-Face       2 Enter News       3 Enterprise User's Group Bulletin     

8. Hány kbájtos ROM-ot tud maximálisan lekezelni a bal oldali ROM-BAY cartridge port?

1 32       2 64       3 48     

9. Hány „lába” van a NICK chipnek?

1 72       2 68       3 70     

10. Mikor jelent meg először a kereskedelmi forgalomban a Spectrum Emulator kártya?

1 1987. május       2 1988. május       3 1988. szeptember     

11. Minek a rövidítése a "NADAGUY"?

1 egy szoftvert gyártó cégé  
 2 egy "a" Stúdió-s szerzőpáros monogramja  
 3 egy amerikai űrkutatási kód     

12. Mennyi az angol számítógéphez behelyezett ENTERPRISE PLUS cartridge után fennmaradó szabad bájtok száma EXDOSCARD nélkül?

1 114858       2 116121       3 110477     

13. Milyen kapacitású lemezt nem tud kezelni az EX-DOS 1.0 verziója?

1 360 kbájt       2 1.2 Mbájt       3 720 kbájt     

13+1 Melyik cég a kifejlesztője az Enterprise BASIC Interpreterének?

1 MICROSOFT       2 ENTERSOFT Ltd.       3 INTELLIGENT SOFTWARE Ltd.     

## Megváltozott karakterek

Farkas Attila olvasónk a karakterek át- és visszaalakítására írt programot, amely a karaktergenerátort használja. A lista 147-es sorában szereplő TEXT utasítást szabadon el lehet hagyni. A képernyőn látványosan alakulnak át az Enterprise karakterei.

```
5 PRINT :PRINT "EGY KIS TURELME"
10 DIM A(1152)
20 FOR L=46208 TO 47359
30 LET A(L-46208)=PEEK(L)
40 NEXT L
100 FOR I=46208 TO 47359
110 POKE I,255
120 NEXT I
130 FOR I=46208 TO 47359
140 POKE I,A(I-46208)
147 TEXT
150 NEXT I
160 END
```

## Egy kis botkormányosdi

Bár az EXOS hatékonyan támogatja mindhárom botkormány kezelését a \$SJOY nevű speciális funkcióval (alfunkciókód: 9), alkalmanként jó lehet, ha a körülményes csatornakezelést megkerülve, közvetlenül a portokat írva-olvasva férhetünk hozzájuk. Mivel a belső botkormány szerves része a billentyűzetnek, úgy olvashatjuk le, mint bármely billentyűt. Lévéen az ehhez szükséges információ a Mikroszámítógép Magazin 1988. évi 8. számában a 27. oldalon megjelent, nem is térek ki erre bővebben.

Más azonban a helyzet a külső botkormányokkal. Közvetlen elérésükről, ha jól tudom, nem jelent meg eddig semmi, még az EXOS leírásban sincs megemlítve. A CONTROL aljátokba csatlakoztatott botkormányok leolvasásának előkészítése ugyanolyan, mint a billentyűzettel-olvasásé: a 0B5H (181) port 0-3. bitjeibe kell írni OUT utasítással egy 0 és 9 közé eső értéket. Ha az 0 és 4 között van, a CONTROL1 botkormányt, ha 5 és 9 között, akkor a CONTROL2-t fogjuk olvasni. A 0, illetve 5 esetében a tüzgomb felől „érdeklődünk”, 1 és 6 esetén a fel-, 2 és 7 esetén a le-héztetről. 3 és 8 adja meg, hogy a botkormány balra van-e húzva és 4, illetve 9 azt, hogy jobbra-e. Ezt foglalja össze az alábbi kis táblázat:

	tűz	fel	le	bal	jobb
CONTROL1	0	1	2	3	4
CONTROL2	5	6	7	8	9

Ilyen előkészítés után a 0B6H (182) port olvasása IN utasítással adja meg a kívánt információt. Ha a beolvasott port 0. bitje alacsony (0 értékű), akkor az azt jelenti, hogy a botkormány 0B5H porton beállított funkciója él. Vagyis, ha a 0B5H (181) portra előzőleg 2-t írtunk, a 0B6H (182) port 0. bitjének 0 értéke azt jelenti, hogy a CONTROL1-be csatlakoztatott botkormány lefelé van húzva. Ha a bit 1-es értékű, akkor a botkormány ebben az irányban nyugalomban van.

Ha a botkormány összes jellemzőjére rá akarunk kérdezni, a 0B5H portra sorban ki kell írni a 0-4, illetve az 5-9 értékeket például egy ciklussal, vagy minden kírás után le kell olvasni a 0B6H port 0. bitjét, a gépi kódú lista az elmondottakat szemlélteti. Egyébként a program nem más, mint a \$SJOY speciális funkcióit megvalósító rutin részlete a ROM-ban.

A rutin kezdete (RD\_JOY címke értéke) az angol gépeken a 00EFD7H címen van (0. szgmens, 61399); kétnyelvű gépeken két példányban található meg: először az előbbi címen, másodsor az bővítő szgmens 0E3DEH címen (58334). Az, hogy a bővítő szgmens hányas számot visel, az eltérő hardvermegoldások miatt gépenként különböző. Eddigi tapasztalataim szerint vagy 4 vagy 5. Hogy a kettő közül melyik, azt úgy lehet eldönteni, hogy kiolvasuk, mondjuk PEEK utasítással a 00C5H (179) abszolút című bájttartalmát. Ha ennek értéke 4, akkor a bővítő szgmens száma a gépünkön 5, illetve fordítva.

Most pedig a listáról. A rutinba belépéskor a C regiszter határozta meg, hogy melyik botkormányt fogja leolvasni a funkció. Ha C=0, akkor a belsőt, ha C=1, akkor a CONTROL1-be csatlakoztatott (EXT1), ha C-nek bármilyen más értéke van, akkor a CONTROL2 (EXT2) csatlakozóról kapunk tájékoztatást.

RD_JOY	DEC C JP M,RD_INT	: C regiszter csökkentése eggyel : ha C-ben minusz érték lett, akkor a belső botkormányt kell leolvasni
RD_EXT2	LD B,05 JR Z,RD_EXT1 LD C,B	: a ciklust ötször kell lefuttatni : ha C-ben 0 van : EXT2 esetén 5-től kezd a ciklust
RD_EXT1	LD A,C	: a portra kiírandó érték beállítása
	INC C	: a ciklus következő futásakor szükséges érték
	OUT (0B5H),A	: botkormányállapot beolvasása a portról
	IN A,(0B6H)	

## Mi a manó?

Olvasóink leveleiből megtudtuk, hogy a szegedi Centrum Áruházban több Enterprisé gépet is eladtak 2.0 verziószámú BASIC cartridge-dzsel. Elképzelhető, hogy ez más áruházban is így volt, vagyis hogy erre nem hívták fel a vásárlók figyelmét. A gépkönyv, amit a géphez mellékeltek, a 2.1 verziót tartalmazza. Ezért az újdonsült géptulajdonosok meglepődtek, amikor a „BASIS” nyomozás” során a következőkre derült fény. A 2.0 verzió ismeri ugyan a CALL-függvényhívást, de nem tudja értelmezni, ha gépi kódú program részére helyet foglalunk az ALLOCATE parancssal. Együtt a kettő nem alkalmazható. Így – többek között – nem működik a fejbéállító program sem. Illyenkor „A változónak nincs értéke” hibaüzenetet kapjuk azokra a sorokra, amelyekben a CALL hívás szerepel.

A sztringeket is kényesebben kezeli ez a változat. Például a 2.0 verzió csak az alábbiak szerint megírt sorokat fogadja el:

```
100 LET AS$ = "PRINTER"
110 PRINT AS$(3;7)
```

```
START
INTER
```

A 2.1 verzió ezzel szemben a következőket is elfogadja:

```
100 LET AS$ = "PRINTER"
110 PRINT AS$(3;)
```

START  
INTER

Szegedi olvasóinknak ez sajnos csak három hónap után tűnt fel, de akkor már a gépeket nem cserélték ki. Elkeseredésükben a szegedi Professional Szervizhez fordultak. Ott nagy segítőkészséggel találkoztak: kaptak új, 2.1 verziójú BASIC kártyákat. Bár a szerviz dolgozóit hasonló esettel eddig nem találkoztak, azért elképzelhetőnek tartják, hogy nemcsak ezeket a gépeket adták el 2.0 verziószámú BASIC-kel. Ezért javasoljuk, hogy akik még nem próbálták ki, most gépeikbe a HELP parancsot, es így győződjék meg a BASIC kártyájuk verziószámáról. Sok rejtélyes gondnak tolmácsolhat ezzel. Együttal elmentésükkel a szegedi Professional Szerviz dolgozóinak olvasóink köszönetét.



Geatsch Günterné rajza

RRA  
CCF

: a 0. bit a carry flagbe kerül  
: a carry 1 lesz, ha eddig 0 volt, illetve viszont

RL D

: a bitek összegyűjtése D regiszterbe

DJNZ RD\_EXT1  
LD C,D

: ciklusség  
: C-ben a botkormány 5 állapot bitje

XOR A  
RET

: A=0, azaz nem volt hiba

C regiszter tartalma a program lefutása után: 4. bit magas (=16); tűz nyomva; 3. bit (=8); fel; 2. bit (=4); le; 1. bit (=2); balra; 0. bit (=1); jobbra. Mindenütt a magas értékek (1 értékű bit) jelzik az aktív állapotot! Fordítva, mint a közvetlen portkiolvasásnál.

Mivel ez csak egy részlete a rutinának, az RD\_INT címke nem mutat sehoval. A rutin most következő hexadecimális dumptjában ezért alacsonyabb helyi értékű bájtra helyett NL áll, a magasabb helyett NH. Ezt tetszésünk szerint átirhatjuk a saját belső botkormány-leolvasó rutinunkra vagy a megelőző 0FAH értékű bájttal együtt elhagyhatjuk. Ekor ez a rutin nem lesz képes a belső botkormányt tesztelni. (A címke eredeti értéke angol gépen 0EFFFH, német gépen ugyanez, illetve a bővítő szgmens 0E3F6H.) A dumpban minden érték hexadecimális számként értendő:  
0D FA NL NH 06 95 28 01 48 79 0C D3 B5 DB B6 1F 3F CB 12 10 FA 4A AF C9.

Racsó Tamás

## Hardver

A sorozat alap gondolata — azon a régi felismerésen túl, hogy az elektronika és a számítástechnika elválaszthatatlan egymástól — a következő tapasztalatot summázza. A szoftver — a programok — jelentősége egyre nő, de az is tény, hogy az igazán jó (az adott számítógép nyújtotta lehetőségeket maximálisan kihasználó) programok megírásához a programozónak

rendelkeznie kell alapfokú áramkört hardverismerettel is. Megerősíti ezt, hogy szaporodik az olyan berendezések, mikroprocesszort alkalmazó rendszerek száma, amelyek programvezérelten működnek. Az ilyen rendszerek tervezőinek és fejlesztőinek is szükségük van integrált hardver- és szoftverismeretre.

A cikksorozatnak ebben a részében az eddigiek összegzéseként bemutatunk egy Z80-alapú mikroszámítógépet. Sorra vesszük a hardverelésztés lépéseit, majd azt a környezetet, amelynek segítségével a mikroszámítógép programjai fejleszthetők.

## Z80 alapú mikroszámítógép

Amikor a tervező egy feladatot mikroszámítógép segítségével akar megoldani, szinte teljesen biztos, hogy processzor, EPROM és RAM memória feltétlenül szükséges a rendszerbe. Amiben a legtöbb feladat egymástól hardveresen különbözik, az a feladat által igényelt perifériális egységek száma és fajtája.

A fenti megfontolás alapján célszerű a

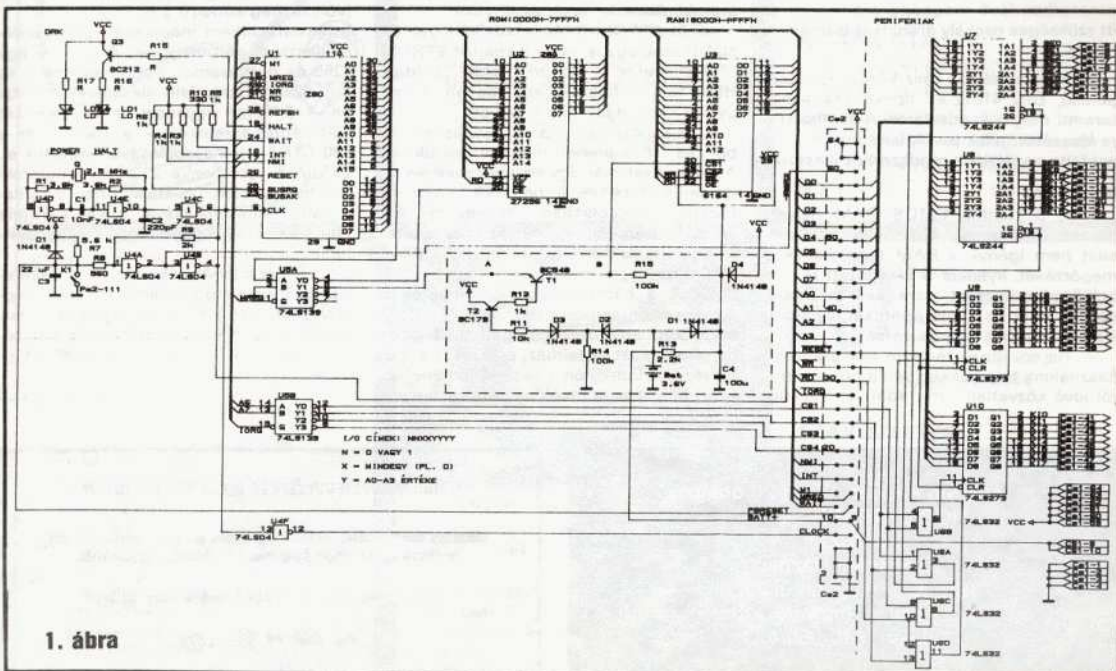
mikroszámítógépet két részre bontani: az egyik a fix rész, amely az állandóan szükséges részeket, a processzort, a memóriákat tartalmazza, a másik pedig, amelyik a feladat-specifikus részeket, a perifériákat foglalja magába. Az így kialakított mikroszámítógép kapcsolási vázlatát az 1. ábrán látható.

A fix rész mindössze öt (!) integrált

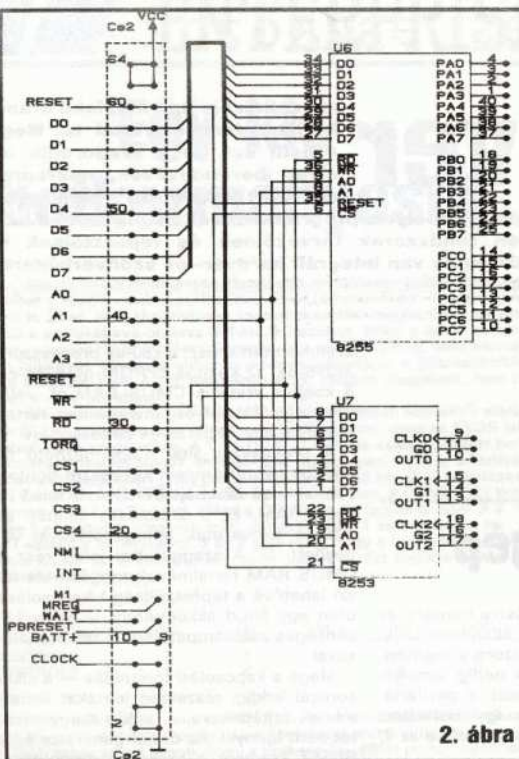
áramkört tartalmaz: a Z80-as processzort, 27256-os, 32 kbájtos EPROM-ot, 6264-es, 8 kbájtos statikus CMOS RAM-ot, valamint egy 74LS04-es, invertereket tartalmazó és egy 74LS139-es címdekódot.

A processzor által megcímezhető 64 kbájtos címterományt két részre bontottuk: az első 32 kbájtos EPROM, a felső 32 kbájtos RAM. Ez a szétosztás csupán az A15-ös címvezeték felhasználásával végezhető el. A szaggatott jelölt rész a CMOS RAM tartalmának megőrzését teszi lehetővé a tápfeszültség kikapcsolása után egy NiCd akkumulátor — vagy közönséges zselblámpatelep — felhasználásával.

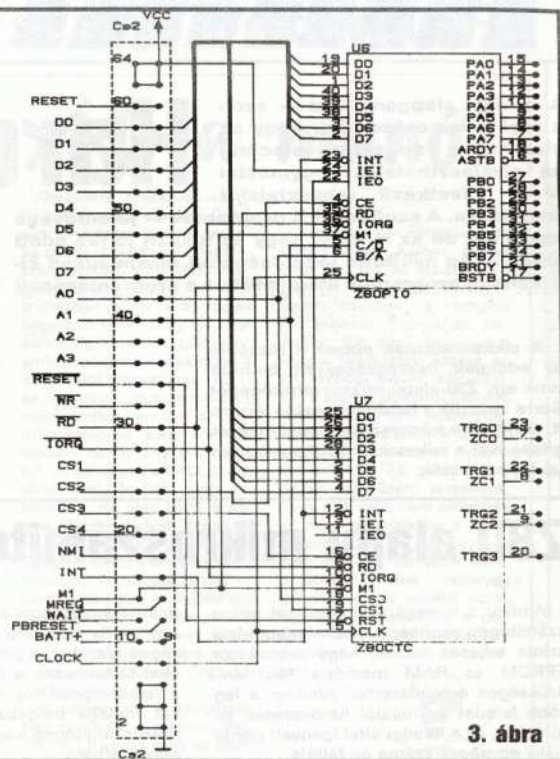
Maga a kapcsolási megoldás — a cikksorozat eddigi részeiben leírtakat ismeretnek feltételezve — külön magyarázatot nem igényel. Az órajel-generátor és a RESET áramkör a szokásos megoldású. A cím-, adat-, és vezérlővonalak külön



1. ábra



2. ábra



3. ábra

meghajtást nem igényelnek, mivel a processzorban lévő meghajtó áramkörök az itt szükséges csekély áramokat biztosítani tudják.

A fix és a változó rész között egy 2x32 pólusú, DIN 41612-es típusú csatlakozó teremti meg a kapcsolatot. A csatlakozóra kivezetett jelek teszik lehetővé, hogy a sokfajta perifériát a rendszerhez illesszük.

#### Megjegyzések:

— A 8 kbájtos CMOS RAM helyett használható normál kivitelű is, ha a feladat nem igényli a RAM tartalmának a megőrzését. Ilyenkor a szaggatott részen belüli alkatrészeket nem kell beültetni, csupán az A és B jelű pontokat, valamint a D4 diódát kell rövidre zárni.

— Ha a változó részben sok áramkört használunk fel, szükség van a processzorral jövő közvetlen cím-, adat- és vezérlő-

vonalak meghajtására, ami buszmeghajtó áramkörökkel lehetséges.

— Az EPROM/RAM részek könnyen átalakíthatók úgy is, hogy 8 kbájtos EPROM (2764), illetve 2 kbájtos RAM (például HM6116) is használható legyen a nagyobb kapacitású tokok helyett.

— A kapcsolásban a perifériális rész 16 be- és 16 kimeneti vonalat tartalmaz. A megoldást már egy korábbi részben — de sajnos hibásan — bemutattuk. Az ott közölt kapcsolásban tévedésből a 74LS244 áramkör engedélyező bemenetei úgy tételeztük fel, hogy egymással VAGY kapcsolatban vannak, és így engedélyezik a háromállapotú bemeneteket. Az áramkör ténylegesen úgy épül fel, hogy két, egymástól független, 4 bites meghajtórészt tartalmaz, és ezeket engedélyezik külön-külön a vezérlőbemenetek. Ezért helyesen is, a tokok engedélyező jeleit

külső VAGY kapukkal kellett kialakítani (volt is még kettő...).

Illusztrációként még két további perifériaillesztést mutatunk be. A 2. ábra egy 8255-ös párhuzamos perifériaáramkör és egy 8253-as számláló illesztést mutatja be. A 3. ábrán ugyanezt az illesztést két Z80 perifériaáramkör — a Z80-PIO és a Z80-CTC — felhasználásával végeztük el. Ez utóbbi esetben a Z80-as rendszerekben használatos, ún. Daisy Chain prioritási lánc felhasználásával a két periféria megszakítási működéssel is üzemeltethető.

A rendszertervezéskor az adott feltételeknek megfelelő perifériális részek megtervezése után a nyomtatott áramköri tervezést csupán ez utóbbi részre kell elvégezni, mivel a fix rész változatlan és független.

Dr. Kónya László

Minden kedden 17-től 20 óráig  
HCC ENTERPRISE klub  
a VSZM  
Közösségi Házban  
(Bp. XI., Fehérvári út 120.)  
Klubvezető: Romvári Gábor  
Telefon: 810-950/473

#### A TUDOMÁNSZERVEZÉSI ÉS INFORMATIKAI INTÉZET

előzetes megbeszélés szerint díjmentes programbemutatót tart (vidéken is) az általa forgalmazott oktatóprogramokból.

Horváth Zsuzsa 665-011/2663 mellék vagy 813-197

Budapest, Pf. 454, 1372

# BÖRZE.



## COBRA COMPUTER

ELEKTRONIKAI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET  
1087 Budapest IX., Illetos út 7. Telefon: 476-160/388

SZÁMLAKÉSZÍTÉS  
 A KÖNYVELESIG

COBRACONTO

ügyviteli programrendszer moduljai:

- számlakészítő,
- számlanyilvántartó,
- bér- és jövedelemszármefítő,
- főkönyvi könyvelő,
- anyagnyilvántartó,
- általános könyvelési programok.

KÉRJEN RÉSZLETES TÁJÉKOZTATÓT!

## TUTTI

### ELECTROCOOP

KISSZÖVETKEZET

- Eredeti számítógépek
- Kompatibilis XT/AT, 386 konfigurációk
- Szünetmentes tápegységek
- Egyedi megrendelésfelvétel

Cím: Bp., Üllői út 81. 1091  
Tel.: 334-354  
Telex: 22-7230  
Telefax: 149-869



## ENET



Lokális hálózat  
IBM kompatibilis gépekre  
Külön hardver  
nem szükséges  
Ára gépenként:  
19 900 Ft + áfa  
ECOSOFT KISSZÖVETKEZET  
Tel.: 863-677

## ASY

ELEKTRONIKA

A legújabb ajánlatunkból:

- ASY-16 SZUPERMIKRO számítógépek (12 terminál, UNIX operációs rendszer)
  - IBM PC/XT-AT-kompatibilis számítógépek
  - megrendelő által definiált betűkészlettel rendelkező billentyűzetek
  - elektronikai elemek (integrált áramkörök, ellenállások, tranzisztorok)
  - monitordobozok, műszerházak
  - szoftvertermékek és fejlesztések
- KERESKEDELMI IRODA:  
1061 BUDAPEST  
LISZT FERENC TÉR 10.  
TEL.: 415-166, TELEX: 22-4378

## ARECO KFT.

a Mikropo KSZ fejlesztési témáinak jogutódja, felajánlja szabad fejlesztőmérnöki kapacitását. ARECO Informatika KFT.  
Tel.: 427-453



## PERIFÉRIA



Elektronikai  
Fejlesztő  
és Szolgáltató  
Kisszövetkezet  
Bp. VII., Petyerdy u. 30.  
Telefon: 213-588

### AJÁNLATA:

- P-XT: 140 E Ft-tól + áfa
  - P-AT: 200 E Ft-tól + áfa
- igény szerinti konfigurációk
- FX-1000 printer: 75 E Ft + áfa
  - 40 MB-os WINCHESTER:  
86 E Ft + áfa

## procontrol



IRÁNYÍTÁSTECHNIKA  
KISSZÖVETKEZET

MÁRKABOLT, BEMUTATÓTEREM  
62/21-165, 28 985  
Szeged, Kazinczy u. 8. Tel.: 12-259  
Telex: 82-726

386 AT 25 MHz toronygép	299 000 Ft-tól
286 AT 16 MHz TURBO AT gép	98 000 Ft-tól
- 86 XT 12 MHz	
TURBO XT gép	69 000 Ft-tól
LAPTOP hordozható	
LCD XT gép + modem + akku	149 000 Ft
12" zöld monochrom monitor TUV M31	4 900 Ft
5 1/4" DS DD Floppy lemez	79 Ft

és még 2000 árucikkről árjegyzék, azonnali kiszolgálás.



**BROTHER AX 15 típusú,**  
margarétafejes,  
elektronikus írógép

Megvásárolható:  
Budapest VI.,  
Népköztársaság útja 2.  
Tel.: 531-231



1146 Bp.,  
AJTÓSI DÜRER SOR 10.  
Levélcím:  
1393 Pf.: 319.  
Telefon: 421-974  
Telex: 22-6544

### SZAKÜZLETEK:

1. Bp. VI., Szinyei Merse u. 1.  
Tel.: 127-628. Tx.: 22-6684
2. 9022 Győr, Lukács S. u. 18.  
Tel.: 96-14880. Tx.: 02-4679
3. 3100 Salgótarján, Ady Endre u. 1.  
Tel.: 32-10971. Tx.: 22-9380

## Prológ

Noha mind szélesebb körben terjednek az IBM PC kompatibilis gépek, még egyetlen hazai folyóirat sem vállalta, hogy az ezekhez szükséges gyakorlati hardver- és szoftverismereteket közreadja. Bár lehet, hogy a más gépek tulajdonosait szándékomon kívül megsértem, de hittél valom, hogy a PC-kategóriába tartozó számítógépek napjainkban a felhasználók igazi munkaeszközei.

A „Bécsi út másik végén” uralkodó árak lehetővé teszik, hogy ezek a gépek a kisebb pénzü magánfelhasználóknak is elérhetővé váljanak.

A rovatban különféle témájú, mélységű és formájú cikkek kapnak helyet. Fórumot kívánunk teremteni a gépekkel most ismerkedőknek, illetve az azokat már alkalmazóknak. Nem akarjuk élesen elkülöníteni a hardvert a szoftvertől, mert véleményünk szerint csak a számítógéppel megoldandó feladat van, amihez a gép szoftver- és hardverlehetőségeit maximálisan ki kell használni. Az viszont tény, hogy igazán jó, gyors és hatékony programot csak a számítógép hardverjének ismeretében írhatunk.

Alapelveként fontosnak tartjuk, hogy a közötti írások a gyakorlatban hasznosak, tág ismereteket nyújtóak legyenek. Egy-egy témáról nemcsak általánosságban, „nagyvonalúan” írunk, hanem úgy, hogy a közölt ismereteket azonnal használni is lehessen. Tudjuk: a problémákról nemcsak beszélni kell, hanem meg is kell oldani azokat!

A cikkek között éppúgy lesznek a gépvásárlást vagy bővítést segítő termékismertető, mint a felhasználást könnyítő írások. Használati útmutatókat is közlünk, mivel a vásárolt programok többsége és a hozzájuk tartozó dokumentációs anyagok angol nyelvek. Ezért is célszerű az alkalmazást segítő tömör, lényegi fordítások közreadása.

Nem célunk viszont, hogy kizárólag „eredeti”

anyagokat publikáljunk. Bátran nyúlunk a nyugati folyóiratokban már megjelent témákhoz.

Eredetileg nem építettünk arra, hogy az olvasók majd cikkekkel segítik a rovatot, de azért szívesen látjuk a színvonalas írásokat!

Teret adunk az úgynevezett szabad forgalmú (public domain) szoftverismertetésnek, bemutatásnak is. Az e kategóriába tartozó programok szabadon terjeszthetők, felhasználhatók. Külföldön erre kialakították az infrastruktúrát: a telefonvonalon elérhető adatbankokat. Sajnos arra, hogy a programokat lapon keresztül terjesszük — a korábbi szomorú tapasztalatok birtokában —, csak szűk lehetőséget látunk. Elképzelésünk szerint a postán, utánvéttel küldött mágneslemezek terjesztésére vállalkozunk majd. Az el-lenszolgáltatásért kért összegnek — nyereség nélkül! — fedeznie kell a lemez, a csomagolás, a feladás, a lemezmasolás és az ezekre fordított munkaidő költségeit.

Mivel a rovatvezető munkahelye a Kandó Kálmán Villamosipari Műszaki Főiskola, ezért a hallgatók témakörhöz kapcsolódó munkáit is bemutatathatjuk. A rovat fő tématerületei a műszaki kutatáshoz, fejlesztésekhez kapcsolódnak.

Ízelítőnek — és kedvcsinálóknak — néhány témakör- és cikkleírást közreadunk:

**Alkalmazások** — speciális témájú cikkek

**Programismertető** — egy-egy IBM-program bemutatása

**Hardver** — IBM PC XT/AT hardverrel kapcsolatos írás

**Parancs-összefoglaló** — egy-egy program parancskészletének összefoglalása

**Ajándék** — a lap olvasóinak egy rövid PC-program

Dr. Kónya László

## Norton Editor

Az IBM PC számítógépen futó sok-sok szövegszerkesztő közül talán az egyik legnépszerűbb és a legtöbbek által használt program a Peter Norton Computing cég által forgalmazott Norton Editor. Az eredeti hirdetés szerint Peter Norton is ezzel a szövegszerkesztővel írta és írja a programjait. Ez a szövegszerkesztő programirási célokra készült, a parancskészlete is elsősorban ezt a célt szolgálja. Mérete mindössze 30 kb-át. A gyors működés érdekében a szerkesztendő fájl a memóriában helyezkedik el, ezért a szöveggel kapcsolatos műveletek nagyon gyorsak.

Mivel a rendelkezésre álló teljes memóriatartományt felhasználja, ezért nagyméretű fájlokat kezelhet. Ha a szerkesztendő fájl nem fér be egyszerre a memóriába, szerkeszthető az egyes részek egymás után is. A szerkesztő parancskészletének csoportjai

az F1-F9 funkciógombokkal aktivizálhatók, majd egy második billentyű megnyomásával választható ki a csoporton belüli parancs.

A program megvásárlásakor magyar nyelvű leírás nem mellékelnek, ezért a következőkben röviden összefoglaljuk a használatával kapcsolatos alapvető ismereteket, és a parancsok magyar nyelvű összefoglalóját is közöljük.

A program hívása:

NE < +xxx> <input fájl> <output fájl> <képernyő paraméter> ahol

— +xxx annak a bemeneti fájlsornak a száma, amelyre be kell állnia a szerkesztőnek (megadása nem kötelező);

— input fájl = a szerkesztendő fájl (a teljes útvonal megadható, ha az nincs kijelölve, akkor az aktuális könyvtárban keresi);





## A Norton Editor (NE) parancskészlete

— output fájl = ahova tárolni akarjuk a megszerkesztett fájlt (ha nem adjuk meg, akkor a bemeneti fájlnev alatt rakja le. Megadásakor a teljes útvonal megadható, ha az nincs kijelölve, akkor az aktuális könyvtárban keresi);

— képernyő paraméter = /DA vagy /DB vagy /DC (megadása nem kötelező). Ezzel az utóbbi paraméterrel tulajdonképpen a képernyő kezelésének a módját adhatjuk meg;

— /DA = ha a képernyőmeghajtó száz százalékos IBM kompatibilis (a képernyő közvetlenül a képernyő-memóriába való írással kezelhető);

— /DB = ha a számítógép BIOS kompatibilis (a képernyő a BIOS hívásokkal kezelhető);

— /DC = ha a számítógép ANSI.SYS képernyő-meghajtóval kompatibilis (a képernyő az ANSI.SYS meghajtóval kezelhető).

A leggyorsabb a DA, utána a DB mód. A leglassabb a DC mód, ami használat közben is jól megfigyelhető.

Hívás után a szerkesztő bejelentkezik, és bármilyen billentyű leütésére dolgozni kezd. Ha híváskor nem adtunk meg bemeneti fájlt, akkor ezt most a bejelentkezőskor fogja kérni. Ha a bemeneti fájl létezik, akkor a képernyőn megjelenik az első oldal, illetve ha +xxx paraméterrel hívtuk meg, akkor az azazal a sorral kezdődő oldal.

A képernyő alsó sorában gyors információ látható a kurzor aktuális helyzetéről, ami mutatja annak a sornak (Line=...), illetve annak az oszlopnak (Col=...) azt az értékét, ahol a kurzor van. Ugyanitt jelenik meg a szerkesztett fájl neve, valamint az, hogy beszúrás (Insert) vagy felülírás (Overstrike) üzemmódban van a szerkesztő.

A szerkesztés közben bármikor az F1-es billentyű lenyomásával kérhetünk segítséget (Help). A segítséget nyújtó képernyőből bármelyik billentyűvel visszatérhetünk a szerkesztendő szöveghez. Ennek az angol nyelvű helpnek a fordítását közöljük a külön táblázatban.

Az editor lehetővé teszi, hogy két fájlt egy időben szerkesszünk. A képernyőn bármikor nyithatunk egy ablakot, ahová egy másik fájlt betölthetünk, illetve szerkeszthetünk. Ebben a másik fájlban kijelölhetünk egy blokkot, amit átmásolhatunk az eredeti szövegbe. Így több fájlból készíthetünk egyet. Az eredeti fájlhoz hozzákapcsolhatunk (Append) egy másik fájlt is. A szerkesztendő fájl memóriafoglalására vonatkozó információ az F2 billentyű lenyomása után jelenik meg a képernyőn. Ha a szerkesztés közben végre akarunk hajtani egy DOS utasítást, ezt az F9 billentyű megnyomásával tehetjük meg.

A szövegszerkesztő mintaszerűen kialakított, gyorsan megtanulható, jól használható programozói eszköz.

### KURZORMOZGATÓ PARANCSSOK

- <- - Kurzor balra
- > - Kurzor jobbra
- ↑ - Kurzor fel
- v - Kurzor le
- ^<- - Kurzor egy szóval balra
- ^>- - Kurzor egy szóval jobbra
- HOME - Kurzor a sor elejére
- END - Kurzor a sor végére
- PgUp - Kurzor a lap tetejére
- PgDn - Kurzor a lap aljára
- ^HOME - Kurzor a fájl elejére
- ^END - Kurzor a fájl végére

### TÖRLŐ PARANCSSOK

- <- - Baloldali karakter törlés
- DEL - Jobboldali karakter törlés
- ^M - Kurzortól balra lévő szó törlése
- ^W - Kurzortól jobbra lévő szó törlés
- ^L - Törlés a kurzortól a sor elejéig
- ^L - Törlés a kurzortól a sor végéig
- ^K - Sortörlés
- F4 D - Blokkötörlés
- ^U - Törölt szöveg visszaállítása

### KÉPERNYŐFORMATUM VEZÉRLÉS

- F5 L - Sortiegyenlítéshez sorhossz
- F5 M - Sortiegyenlítés ki/be
- F5 F - Egy bekezdés formálása
- F5 T - Tabulátor beállítás
- F5 C - Kurzortípus állítás
- F5 D - Képernyőszin állítás
- F5 I - Beljebb kezdés váltó
- F5 S - Editor kialakítás megőrzése

### NYOMTATÓ PARANCSSOK

- F7 P - Teljes szöveg nyomtatása
- F7 B - Blokknyomtatás
- F7 E - Uj lap
- F7 S - Lapméret beállítás nyomtatáshoz
- F7 M - A nyomtatás bal széle beállítás

### KERESŐ PARANCSSOK

- ^F - Sztring keresés előre
  - ^F - Sztring keresés visszafelé
  - ^C - Előre keresés folytatása
  - ^C - Visszafelé keresés folytatása
  - F4 F - Blokkjelölő keresés előre
- Ha azt akarjuk, hogy a sztringekben lévő kis és nagybetűk számítsanak, ESC-t használjunk RETURN helyett.

A kereső sztringbe CTRL+RET új sor karakter keresést jelent.

^C - folytonos keresés előre

### FÁJL PARANCSSOK

- F3 E - Mentés és kilépés
- F3 S - Mentés kilépés nélkül
- F3 Q - Kilépés mentés nélkül
- F3 N - új fájl szerkesztése
- F3 X - Aktív fájl váltás
- F3 M - Szöveg beillesztés kurzortól
- F3 L - Betöltés a fájlt
- F3 A - Két fájl egyesítése
- F3 C - Kimeneti fájl zárása

^ = CTRL bill. \ = ALT bill.

### BLOKK PARANCSSOK

- F4 S - Blokk jelölés
- F4 R - Blokk jelölés törlés
- F4 F - Blokk jelölés keresés
- F4 D - Blokk törlés
- F4 C - Blokk másolás
- F4 W - Másolás másik fájlba
- F4 M - Blokk áthelyezés
- F4 L - Sorrelékelés (soreleméssel)
- F4 E - Sorjelöl. (nem sorelem.)

### TOVABBÍ PARANCSSOK

- F1 - Segítség !
- F2 - Editor státusza
- F9 - DOS parancsok
- INS - Beillesztés (inzer)
- F6 INS - Felülírás
- F6 G - Adott számu sorra lépés
- F6 M - Egyezés keresés: ((I ))
- F6 C - Tömörített kijelzés
- ^P - Vezérlő karakter beírás a szövegbe
- ^V - k/M váltás sor kezdetig
- ^W - k/M váltás sor végéig (k=kis, M=NAGY betű)

### KERESÉS ÉS HELYETTESÍTŐ PARANCSSOK

- Keresés és csere előre:
- ^F keresett sztring beírása
  - ^F helyettesítő sztring beírása
  - Keresés és csere visszafelé: ^F keresett sztring beírása
  - ^F helyettesítő sztring beírása
  - Keresés és csere érvényesítés: Y-cserél
  - N-nem cserél
  - I-mindenhol cserél
  - SPACE-kilépés a cserebeállítás
  - ^C - folytonos keresés vissza



## SZABAD SZOFTVER

# A PKARC tömörítőprogram

Manapság sok szó esik a számítógépes programok és adatok tömörítéséről; arról a módszerről, amivel a fájlok alkotó bájtsorozatát nem eredeti formában, hanem valamilyen tömörítési móddal tároljuk (becsomagoljuk), majd futás előtt eredeti tartalmát visszaállítjuk (kicsomagoljuk). A tömörítés hatékonysága a módszertől és a tömörítendő fájl szerkezetétől függ. Szövegfájlok esetében általában 40 százalékos körül érték.

Azért, hogy fogalmunk legyen a tömörítési módszerekről, röviden közreadjuk kettőnek a lényegét.

Ha a bájtsorozatban egymás után több egyforma bajt van, célszerűbb ezeket darabszám-bajt formában tárolni.

Tömörítési mód a fájlban szereplő bajtok előfordulási gyakorisága szerinti kódolás is. Ilyenkor a bajtokat nem az egyforma 8 bites hosszban tároljuk: a tárolási hosszuk a fájlbeli gyakoriságuktól függ. A leggyakoribb bajtok egy-két bites kóddal, a ritkébbak hosszabb kóddal tárolódnak, a Morse-kódhoz hasonlóan. Például az angol ábécé leggyakoribb betűjének, az e-nek a kódja csak egy pont, ez a leg-rövidebb kód.

Természetesen az adatok tömörítése programmal valósul meg. A hazai terméknek ílyt egyelőre hiába is keresnénk. A forgalomban lévő másolt szoftverek között viszont sok helyen megtalálható az ARC program valamelyik verziója.

A most bemutatandó PKX35A35 programcsomag „szabad” szoftver. Ezért lelkesen használhatjuk Philip Katz gyors és biztonságos archiváló programját. Ennek sok kitűnő tulajdonsága mellett talán leg-kedvesebb jellemzője a *legális* volta.

Az archiváló rendszer három programból áll: a kicsomagoló PKARC programból és az EXE fájlba való becsomagolást végző PKSFX programból. A következőkben röviden ezeket a programokat mutatjuk be. A leírás azonban csak kivonat, a leg-szükségesebb tudnivalókat tartalmazza. A programból, azaz a PKX35A35.EXE fájl-ból kicsomagolható az eredeti angol nyelvű leírás is.

### PKARC Ver. 3.5 — fájlok tá-mörítő program

Az archiv (tömörítő) szoftver egyetlen fájlba több fájlt is tömörít. A fájl kiterjesztése minden esetben .ARC, erről ismeri fel a program. A tömörített fájlok kevés helyet foglalnak el a lemezen, rövid idő alatt továbbíthatók vagy menthetők. Az arc-íváló program az adatállományt analizálja és optimális eljárással tömöríti azt. Az archiv fájl a tömörített adatok biztonságos visszakeresésére saját 16 bites ciklikus redundanciátáblát (CRC) tartalmaz.

#### Szintaxis:

PKARC [kompatibilitási opc.] archiv opció [fájlnev ...]

#### Kompatibilitási opciók:

A programmal régebbi verzió generálta fájlok és e rendszer kompatibilitását lehet elérni.

#### Lehetséges archiv opciók:

a = fájl hozzáadása  
f = fájl felfrissítése  
u = archiv állomány update-olása  
l = szoftverlicenc kiírítása  
x = egész állományra vonatkozó  
g <password> = titkosított fájl(ok) kulcsszava

d = fájl törlése  
m = fájl mozgatása (move)  
v = kommentek csatolása  
c = új/más kommentek bevitelle  
Komment = a terminológia szerint az egyes archiv állományokhoz a felhasználó által hozzáfűzhető megjegyzés.

Password = az egyes titkosítási eljárások során a felhasználható kulcsszó. Ezt a felhasználó választja és csak ennek ismeretében állítható helyre az állomány. A helyreállításához ismerete elengedhetetlen, a programból nem fejthető vissza.

A PKARC a batch fájlban is használható. Ha az eljárás eredményes volt, nulla hibaszinttel (errorlevel) lép vissza a DOS-ba. Ha bármilyen probléma adódott, akkor a hibaszint nullától eltérő.

### PKXARC — tömörített fájlok tá-visszaállító program

Az archiv fájl kiterjesztése kötelező érvényenel .ARC. A program a tömörített állományból való helyreállításra szolgál. A PKXARCJR-nak mindazon lehetőség adott, mint a PKXARC-nak, de jóval kisebb az operatívár-igénye, és ugyanakor lassúbb is annál.

#### Az utatás szintaxisa:

PKXARC [opciók] tömörített fájl [d:\patch] [fájl ...]

#### Lehetséges opciók:

- r fájl(ok) helyreállítása
- c fájl kiírnyítása a képernyőre (helyreállítás nélkül)
- t archiv állomány tesztelése
- e, -x fájl(ok) kiterjesztése eredeti mére-tre
- g <password> kulcsszóval védett fájl(ok) kicsomagolásának engedélyezése
- v kommentált archiv lista

#### PACK.BAT

```

echo off
c:\archiv\pkarc a c:\archiv\%1
if errorlevel 1 goto NEV
echo on
del *.*
echo off
cls
echo *****
echo * Elkészitem az archivált állomány tartalomjegyzékét? (I/N) *
echo *****
in
if errorlevel 2 goto END
c:\archiv\pkxarc -v c:\archiv\%1 > %1.dir
goto END
:NEV
cls
echo *****
echo * A program helyes használata: pack [könyvtárnév]! *
echo *****
:END
    
```



- p fájl kiírnyitása a printerre (helyreállítás nélkül, csak text-fájl)
  - l szoftverhasználati engedély
- Az egyes utasításokban a DOS megszokott „jokerei”, mint a \* vagy a \*.\* és a ??? alkalmazhatók. Az archiv fájl kiterjesztése minden esetben .ARC.

d\path=kijelölt meghajtó vagy útvonal. A meghatározott útvonal-specifikációt mindenképpen egy vagy több szóközzel kell elkülöníteni azoktól a fájloktól, amelyeket kijelöltünk.

fájl=név (nevek), amelyekkel valamit csinálni akarunk (például szöveg helyreállítása stb.).

**Az opciók magyarázata:**

- r Amikor ezzel az opcióval csomagolunk ki és az aktuális alkönyvtárban ezen a néven már létezik a fájl, akkor megkérdezi: overwrite?, azaz felülírjam? Ha y-nal válaszolunk, akkor felülírja, ha n-nel, akkor nem.
- c Ez az opció a képernyőre hozza kicsomagolás nélkül a megadott nevű szövegfájl tartalmát.
- p Helyreállítás nélkül kinyomtatja a megadott fájlok tartalmát.
- t Ellenőrzi, hogy a belső CRC alapján helyreállíthatóak-e a tömörített fájlban levő állományok. Ha igen, a fájl nevének, valamint egy OK üzenetnek a közlésével nyugtázza. Ha \*.\*-ot adunk a fájl kijelölésekor, akkor az egész archivumot teszteli, különben csak a megadott fájlt.
- v Ez az opció a kommentált lista formájában jeleníti meg a tömörített fájl tartalmát: fájlnev, eredeti hossz, tömörítési mód, tömörített hossz és a helynyereség százalékban kifejezve.
- l Megjeleníti a programkészítők használati engedélyét.
- e, -x Ezek az opciók a SeaWare Inc. ARC programrendszerével biztosítják a kompatibilitást.
- g <password> A jelszóval titkosított fájl helyreállítását engedélyezi akkor, ha a PKARC használatakor a megadottal azonos jelszót közöltünk.

Ha a PKXARC-ot .BAT állományokban használjuk, akkor a PKXARC a hibától függően különböző DOS hibaszinttel (errorlevel) hagyja ott a folyamatot, amelyre azután a programokban hivatkozni lehet. Ha a megadott opcióval a program zavartalanul futott le, akkor a PKXARC visszaterési (return) kódja, azaz hibaszintje minden esetben opciótól függetlenül "0"

```
UNPACK.BAT
echo off
c:\archiv\pkxarc c:\archiv\%1
if errorlevel 1 goto NEV
del c:\archiv\%1.arc
if exist %1.dir del %1.dir
goto END
:NEV
cls
echo *****
echo * A program helyes használata: unpack [könyvtárnev]! *
echo *****
:END
```

**PKSFX — önkicsomagoló modul**

A PKSFX.PGM modul segítségével létrehozható az önkicsomagoló .EXE fájl. Ez a programfej teljesen szabadon átnevezhető.

**Az önkicsomagoló program létrehozásának lépései:**

- 1. Az aktuális könyvtárban a MAKESFX.COM elindításával születik a PKSFX.PGM fájl.
- 2. A PKARC segítségével összecsomagolt, tömörített állomány elé másoljuk be a PKSFX.PGM-et. Az utasítást az alábbi szintaxis (minta) szerint kell kiadni:  
copy/b pksfx.pgm+archive.arc fájl.exe ahol  
— a pksfx.pgm a kicsomagoló szoftverrészt

szükséges COPY utasítás szintaktikáját.

**Az önkicsomagoló .EXE fájl fontos opciói:**

- t Teszteli az archiv állományt, mielőtt elindítjuk a kicsomagolási folyamatot; másolás után célszerű lefuttatni! Ha rendben van minden, akkor DOS promptot ad. Ekkor elindítható! Az archivumban levő szövegfájlokat a képernyőre írja, a kiíratás a CTRL—S-sel pillanatszerűen leállítható, és bármelyik billentyű megnyomásával folytatható. CTRL—C végleg leállítja. Ilyenkor a kiíratásra szánt fájl(ok) nevét meg kell adni. \* (joker) is használható!
- r Egyes kiválasztott fájlok kicsomagolása; itt a DOS szintaktikájának megfelelően teljes útvonalat is meghatározhatunk.
- p Ennek az opciónak a segítségével a

```
AUTOPACK.BAT
echo off
copy /b c:\archiv\pksfx.pgm+c:\archiv\%1.arc %1.exe
if not exist %1.exe goto NEV
goto END
:NEV
cls
echo *****
echo * A program helyes használata: autopack [könyvtárnev]! *
echo *****
:END
```

- az archive.arc lehet bármilyen PKARC állomány
  - a fájl.exe bármilyen elnevezésű lehet, ha a név a DOS szabályainak megfelelő.
- Kiterjesztése viszont kötelezően: .EXE

**A MAKESFX.COM program**

A MAKESFX.COM hozza létre a PKSFX.PGM fájlt. Ennek az aktuális könyvtárban van a helye.

A MAKESFX elindítása után angolul bejelentkezik, majd bármelyik gombot megnyomva létrehozta a PKSFX.PGM-et. Ha ez sikeresen lefutott, rövid angol nyelvű üzenetben közli az .EXE létrehozásához

tömörített szövegfájl az eredeti forma helyreállítása nélkül küldhető a nyomtatóra. Szintaktikája azonos a képernyőre való kiíratással.

- g <password> Segítségével az archiv állományt csak akkor lehet kicsomagolni, ha ismerjük azt a személyes jelszót, amelyet a PKARC-kal való tömörítéskor opcióként megadtak. A kulcszó a kiíratott DUMP állományból nem fejthető vissza!
  - l Megjeleníti a PKX35A35.EXE készítőinek felhasználási engedélyét. Szabad szoftver, használatáért nem szabad pénzt kérni, és az eredeti copyrightot megváltoztatni.
- A program felhasználói szoftverekben alkalmazható!



—h Megjeleníti a helpet, amely angol nyelven összefoglalja a fenti opciókat.

Az önkicsomagoló szoftver — miként a programrendszer többi tagja is — az összehajtogatott állományt sőtétlenül hagyja. Fontos tudnivaló, hogy ha hasonló nevű állományt talál a célkönyvtárban, akkor fájlönként megkérdezi, hogy a meglévő, nem tömörített állományt felülírja-e. Y-nal, azaz yessel kell válaszolni, ha ezt akarjuk, és n-nel, azaz noval, ha a meglévőt nem kívánjuk felülírni. Ezzel a programrendszer azonos nevű fájljai alkönyvtárak nélkül is praktikus cserebe-rellhetők. Valamennyi funkció .BAT állományból is hívható.

A kilépési szint (errorlevel) nulla, ha baj nélkül lezajlott a meghívott folyamat, ellenkező esetben nem nulla.

A továbbiakban a PKARC rendszer egyszerű használatát elősegítő batch programok alkalmazását mutatjuk be. Tétélez-zük fel, hogy a batch programok a C:\ARCHIV alkönyvtárban találhatóak. Alkalma-zásuk két különböző módon lehetséges:

[1] A segédprogramok hívásakor mindig meg kell adni az elérési utat.

[2] Az AUTOEXEC.BAT fájlba beírjuk a PATH C:\ARCHIV; elérési utat, illetve ha a PATH utasítás létezik, ki kell bővíteni azt a C:\ARCHIV; alkönyvtárral

## I. PACK.BAT

A program feladata: egy teljes könyvtár archiválása (becsomagolása).

A program hívása:

- [1] C:\ARCHIV\pack [könyvtárnév]
- [2] pack [könyvtárnév]

A program működése: az aktuális könyvtárban található összes állományt összehajtogatja és elhelyezi a C:\ARCHIV könyvtárban [könyvtárnév].ARC néven, majd felajánlja az aktuális könyvtár állományainak törlését. (Ez tulajdonképpen a helytakarékoság!) Ezután kérésre a be-

## UPDATE.BAT

```
echo off
c:\archiv\pkarc f c:\archiv\%1
if errorlevel 1 goto NEV
if not exist %1.dir goto END
del %1.dir
cls
echo *****
echo * Felfrissitem az archivált állomány tartalomjegyzékét. *
echo *****
c:\archiv\pkarc -v c:\archiv\%1 > %1.dir
goto END
:NEV
cls
echo *****
echo * A program helyes használata: update [könyvtárnév]! *
echo *****
:END
```

csomagolt állományról katalógust készít, amelyet [könyvtárnév].DIR néven az aktuális könyvtárba ír be.

Példa a programra: C:\ARCHIV\pack f188

## II. UNPACK.BAT

A program feladata: egy teljes könyvtár kicsomagolása.

A program hívása:

- [1] C:\ARCHIV\unpack [könyvtárnév]
- [2] unpack [könyvtárnév]

A program működése: az aktuális könyvtár tartalmát állítja helyre, kicsomagolva a C:\ARCHIV könyvtárban található [könyvtárnév].ARC állományt az aktuális könyvtárba, majd az archivált [könyvtárnév].ARC állományt törli a C:\ARCHIV könyvtárból. Végül a [könyvtárnév].DIR állományt törli az aktuális könyvtárból.

Példa a programra:  
C:\ARCHIV\unpack f188

## III. AUTOPACK.BAT

A program feladata: egy teljes archivált könyvtár önkicsomagoló állománnyá alakítása.

A program hívása:

- [1] C:\ARCHIV\autopack [könyvtárnév]
- [2] autopack [könyvtárnév]

A program működése: a C:\ARCHIV könyvtárban található .ARC kiterjesztésű aktuális állományból olyan .EXE állományt állít elő, amely futtatása során — a PKARC programok jelenléte nélkül — képes az eredeti archivált állományok (könyvtár) kicsomagolására. A program a futtatható .EXE állományt a híváskor aktuális meghajtón, illetve annak aktuális könyvtárában az eredeti .ARC állomány törlése nélkül állítja elő.

Használatra különösen akkor előnyös, amikor hajlékony mágneslemezen kell az egyik gépről a másikra nagy mennyiségű adatot, programot stb. áthelyezni, vagy ezekről biztonsági másolatot készíteni.

Példa a programra: C:\ARCHIV\autopack f188

## IV. UPDATE.BAT

A program feladata: egy teljes archivált könyvtári állomány felfrissítése a könyvtárban időközben módosított állományokkal. Használatra csak akkor lehetséges, ha a könyvtár archiválása során (lásd PACK.BAT) annak tartalmát nem törlöttük.

A program hívása:

- [1] C:\ARCHIV\update [könyvtárnév]
- [2] update [könyvtárnév]

A program működése: a C:\ARCHIV könyvtárban található .ARC kiterjesztésű archivált állományt összehasonlítja az aktuális könyvtár tartalmával. Az időközben módosított állományokat archiválja az eredeti archiv állományok felülírásával. Ugyancsak felülírja — természetesen, csak ha létezik — a könyvtár archiv katalógusát.

Példa a programra: C:\ARCHIV\update f188

A Floppy lap nyomán

— KL —

# Iskolaszámítógép-szerviz

1088 Budapest, Rákóczi út 25. Telefon: 381-121

## VÁLLALJA:

IBM PC/AT, IBM PC/XT  
és Commodore típusú (C16, C Plus/4, C64, C128)  
gépek javítását, átalánydíjas szervizét,  
egyéni programok, programcsomagok készítését.



# HARDVER



## Nyolc be- és nyolc kimenetes kártya IBM PC géphez

A széles körben elterjedt IBM PC kompatibilis számítógépek felhasználása és alkalmazása gyakorta igényli a számítógépre helyezhető digitális be/kimeneti kártya tervezését. A következőkben egy ilyen célú egyszerű kártyát mutatunk be.

### Az IBM PC-busz

Az illesztőkártyák az alaplapon lévő kártyacsatlakozók valamelyikébe helyezhetők. A kártyacsatlakozókat párhuzamosan kötötték össze, és az alapkártya mindazon jeleit rákapcsolják, amelyek a perifériák vagy egyéb kiegészítő egységek illesztéséhez szükségesek. Ezt a csatlakozó- és jelrendszert nevezzük IBM PC-busznak.

Az IBM PC-busz 2 x 31 pólusú csatlakozósorból áll. A kiosztását az 1. ábrán mutatjuk be.

A jelek aktív, hatásoz szintjét a jel álló + (aktív magas) vagy - (aktív alacsony) jelzés mutatja. A kivezetések közül

sok jel a memóriához közvetlen hozzáférést és a megszakítások kezelését szolgálja. Számunkra fontosak a címvonalak (A0-A19), az adatvonalak (D0-D7) és a következő vezérlőjelek: AEN, RESET DRV (a továbbiakban: RESET), CLK, I/O CHRDY, I/O READ (IOR), I/O WRITE (IOW). A csatlakozón a tervezett kártya tápellátásához szükséges korlátozott terhelhetőségű tápfeszültségek: +5 V, +12 V, -5 V, -12 V is rendelkezésre állnak. A busz minden jele TTL kompatibilis.

### Programozási alapok

A gépben lévő 8088-as mikroprocesszor IN utasítással olvas a perifériából és OUT utasítással ír a perifériára. A megcímezhető perifériatartomány 64 kb-ot. Az első 256 darab (címük 00H-FFH) közvetlen címzéssel is kiválasztható. Ilyenkor az utasítás második bájta jelöli ki a címet.

A DX regiszter felhasználásával, regiszteres indirekt címzéssel tudjuk a (0000-FFFF) teljes B/K címtartományt megcímezni. A DX regiszter tartalmazza a kiválasztandó periféria címet. Az I8088-as processzor 8 és 16 bites B/K portokat kezel. A B/K műveletek egyik operandusa az AL vagy AX regiszter, a másik pedig a megcímezett B/K port. A B/K portok leggyorsabban assemblerben kezelhetők, de természetesen a magasabb szintű nyelvek, mint például a BASIC vagy a Pascal is lehetőséget teremt a portok írására, olvasására.

Igen fontos említést tenni az IBM PC egy korlátjáról. Noha a 8088 mikroprocesszor 16 címvonallal használja a lehetséges 64 k B/K vonal címzésére, a PC csak 10 címvonallal (A0-A9) a B/K dekódolásra, amely 1024 B/K vonal címzést jelent. Az IBM ezt a címtartományt fixen kiosztotta, amit a táblázatban foglaltunk össze.

### 1. ábra

NÉV	B1	A1	NÉV
GND	B1	A1	-I/O CH CK
+RESET DRV	B2	A2	+D7
+5V	B3	A3	+D6
+IORG2	B4	A4	+D5
-5V	B5	A5	+D4
+ORG2	B6	A6	+D3
-12V	B7	A7	+D2
-CARD SLCTD	B8	A8	+D1
+12V	B9	A9	+D0
GND	B10	A10	+IO CH RDY
-MEMR	B11	A11	+AEN
-MEMR	B12	A12	+A19
-IOW	B13	A13	+A18
-IOR	B14	A14	+A17
-DACK3	B15	A15	+A16
-DRG3	B16	A16	+A15
-DACK1	B17	A17	+A14
+DRG1	B18	A18	+A13
-DACK0	B19	A19	+A12
CLOCK	B20	A20	+A11
+IORG7	B21	A21	+A10
+IORG6	B22	A22	+A9
+IORG5	B23	A23	+A8
+IORG4	B24	A24	+A7
+IORG3	B25	A25	+A6
-DACK2	B26	A26	+A5
+T/C	B27	A27	+A4
+ALE	B28	A28	+A3
+5V	B29	A29	+A2
OSC	B30	A30	+A1
GND	B31	A31	+A0

IBM-PC BUSZ

### Az IBM-PC I/O CIMKIOSZTASA

CIM (HEX)	HASZNALJA
000-00F	8237A DMA áramkör
020-021	8259A megszakítás vezérlő áramkör
040-043	8253A Időzítő áramkör
060-063	8255A PIO áramkör
080-083	DMA lapregiszterek
0AX	NMI masz k regiszter
0CX	Fenntartva
0EX	Fenntartva
200-20F	Játék port
210-217	Bővítő egység
220-24F	Fenntartva
27B-27F	Fenntartva
2F0-2F7	Fenntartva
2F8-2FF	Soros vonal (COM2)
300-31F	Prototípus kártya
320-32F	Hard disk
37B-37F	Nyomató
380-38C*	SDLC kommunikáció
380-389*	Bináris szinkron átviteli vonal (2.)
3A0-3A9	Bináris szinkron átviteli vonal (1.)
3B0-3BF	IBM monokróm display/nyomató
3C0-3CF	Fenntartva
3D0-3DF	Színes/grafikus kártya
3E0-3EF	Fenntartva
3F0-3F7	Diszk
3FB-3FF	Soros vonal (COM1)

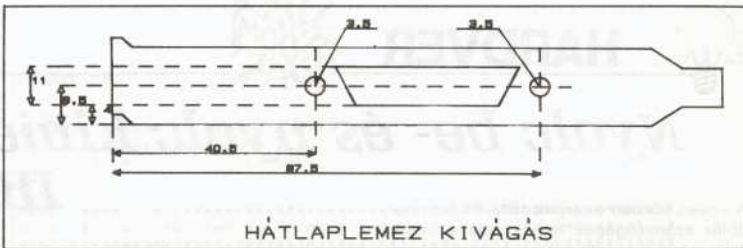
\* Mivel a címek fedik egymást, egyszerre csak az egyiket lehet használni



A táblázatban látható, hogy a címtartomány nagy része foglalt, csupán a 300-31F tartományban található prototípus — azaz az IBM által eredetileg is bővítési-fejlesztési célokra kijelölt — kártyaterület szabad, és ezt a legegyszerűbb felhasználni.

Mivel az IBM PC-ben sok kártya helyezhető el, a PC-busz terhelésének minimalizására gondosan meg kell tervezni a kártyák illesztését. A gyakorlatban ez azt jelenti, hogy kártyahelyenként mintegy két LSTTL egységterheléssel terhelhetjük a buszt. Az adatvonalakat a kártyán feltétlenül buszmeghajtón keresztül használjuk. Ez praktikus egy 74LS245 típusú áramkör, amelynek átgondolt lábelrendezése is nagymértékben egyszerűsíti a kártya áramköri rajzolatának kialakítását.

A cím kiválasztást a címdekódoló áramkör végzi, amely egy digitális komparátor áramkör. Egyik oldalához a cím busz megfelelő jelei, a másik oldalához egy átkötéssor kapcsolódik, amivel beállítható a kártya címe. A legegyszerűbb a 74LS688 típusú, 8 bites komparátorok használata,



HÁTLEPLEMEZ KIVÁGÁS

4. ábra

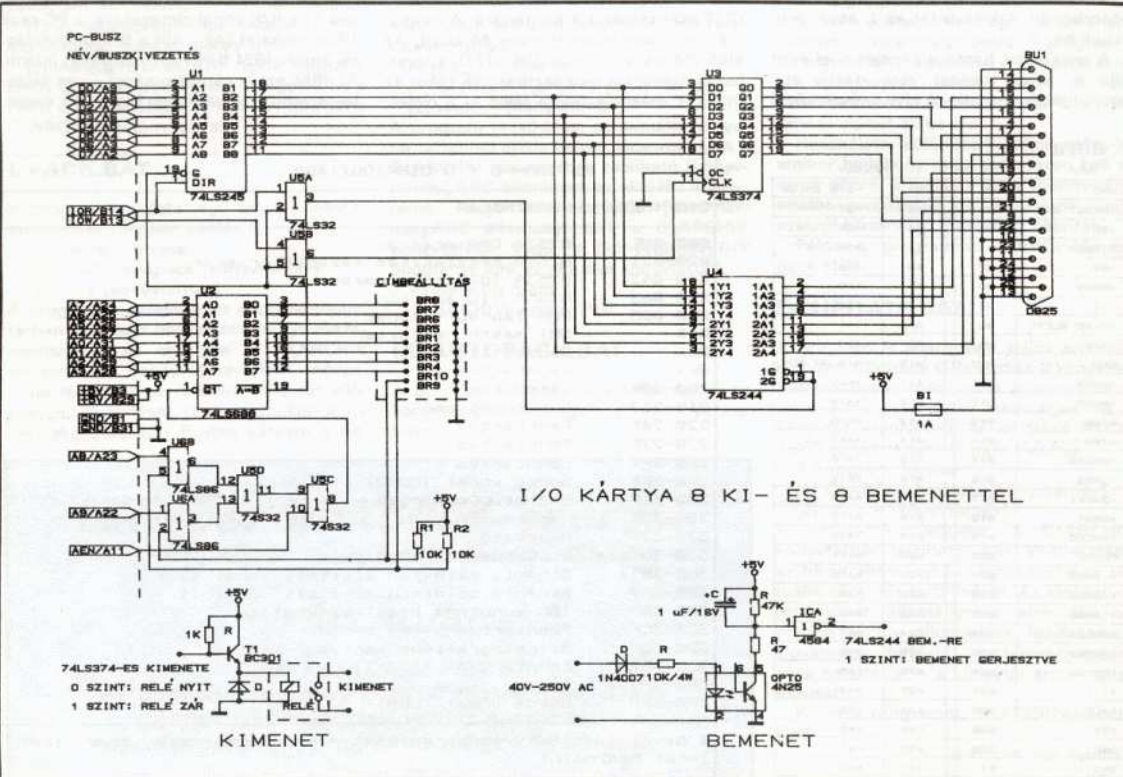
ugyanis a bemeneteiken felhúzó ellenállásokat is tartalmaznak, megkönnyítve a cím kiválasztó átkötéssor illesztését. A címvonalak mellett az AEN jelet szintén a komparátorra kell kapcsolni, mert ennek alacsony szintje jelzi, hogy a kialakult cím érvényes.

Ennyi bevezető után a 2. ábrán bemutatjuk a kártya kapcsolási rajzát.

A kapcsolás PC-buszhoz csatlakozó része a már leírtak szerint alakul. A 300H cím beállításához a BR9 és BR10 jelű át-

kötéseket szabadon hagyjuk, a többi a földre kötjük. A kiírt kívánt bájt U3 (74LS374) tárolóba való írása a címtálat A=B jele, és a busz IOW jele együttes alacsony szintjén következik be. Hasonló módon kapuzódik a bemeneten lévő bájt értéke az U4 (74LS244) áramkörön keresztül, de itt az IOW jel szerepét az IOR jel veszi át. A külvilág felé a csatlakoztatást egy DB25-ös, IBM PC-kben szabványos, 25 pólusú csatlakozó garantálja. A BI jelű biztosítón keresztül az IBM PC tápegység-

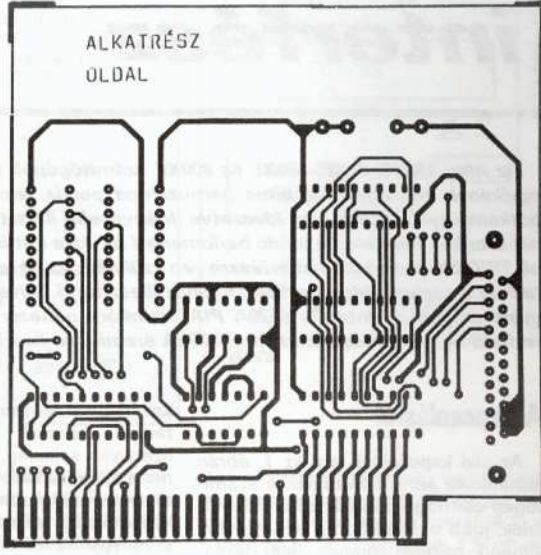
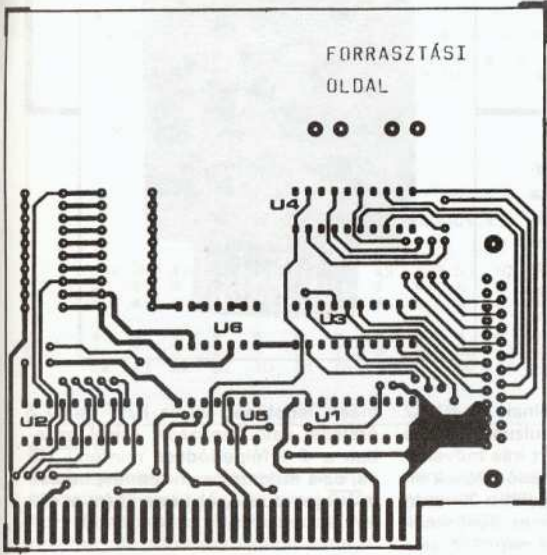
2. ábra



I/O KÁRTYA 8 KI- ÉS 8 BEMENETTEL

KIEMENET

BEMENET



3. ábra

géből maximum 1 amperes áramot használhatunk fel.

A külvilág felé a TTL szintű, 8 digitális be- és kimeneti jel nem felel meg mindig. A 2. ábra alsó felén lévő kapcsolások arra mutatnak példát és követhető megoldást, hogy galvanikusan miként válasszuk le ezeket a jeleket. A kimenetek reléket hajtanak meg, a bemenetek pedig optocsatolón kapcsolódnak, lehetővé téve a nagyobb feszültségű váltóáramú jelek érzékelését.

A könnyebb kapcsolás kedvéért még

két ábrát közlünk. A 3. ábra a kapcsolás kétoldalas nyomtatott áramköri rajzát mutatja. Az alkatrészek elhelyezését ábrázoló ültetési rajzot a kártya egyszerűsége miatt nem is közöljük. A kártyára szerelt hátlaplemez rögzíti a kártyát a számítógépbe. A DB25-ös csatlakozóhoz szükséges hátlaplemez kivágásának mechanikai méretei a 4. ábrán láthatók.

A készre szerelt kártyát a PC-be dugás előtt gondosan ellenőrizzük — különösen a PC-busz felé —, a zárlatok miatt. A busz zárlata ugyanis tönkretelheti a PC-alapla-

pon lévő áramköröket. Az alaplap javítása pedig nem gyerekjáték! Célszerű külső tápegységgel a kártya áramfelvételét még a kiemelt állapotban megmérni. Ez néhány tíz mA lehet csupán.

A kártyát valamelyik csatlakozóba dugjuk, és a gépet bekapcsoljuk. Ha a rendszer feléled, akkor túl nagy baj már nem lehet. Ezek után az 5. ábrán bemutatott BASIC vagy a 6. ábrán látható Pascal programmal ellenőrizhetjük a kártya működőképességét.

K.L.

5. ábra

```

100 REM TESZTPROGRAM AZ I/O KÁRTYÁHOZ
110 REM KIMENET ELLENŐRZÉS
120 REM
130 B=0
140 PRINT "KÉREM A BINARIS ÉRTEKET"
150 INPUT S$
160 IF LEN(S$)>8 THEN PRINT "NYOLC 0-1
SZÁMOT KELL!" GOTO 140
170 FOR I=0 TO 7
180 IF MID$(S$,I+1,1)="1" THEN B=B+2^(7-I)
GOTO 210
190 IF MID$(S$,I+1,1)="0" THEN 210
200 PRINT "CSAK 0 ES 1 SZÁMOT LEHET
GÉPELNI!":END
210 NEXT
215 PRINT B
220 OUT $H300,B
230 END
300 REM-----
310 REM BEMENET ELLENŐRZÉS
320 REM
330 CLS:LOCATE 8,30
340 PRINT "A BEMENET ALLAPOTA:"
350 E=INP($H300)
360 E$=""
370 FOR I=0 TO 7
380 IF (E AND 2^(7-I))>0 THEN E$=E$+"1":
GOTO 400
390 E$=E$+"0"
400 NEXT
410 LOCATE 10,30,0
420 PRINT E$
430 IF INKEY$="" GOTO 350
440 LOCATE ,,1
450 END
  
```

6. ábra

```

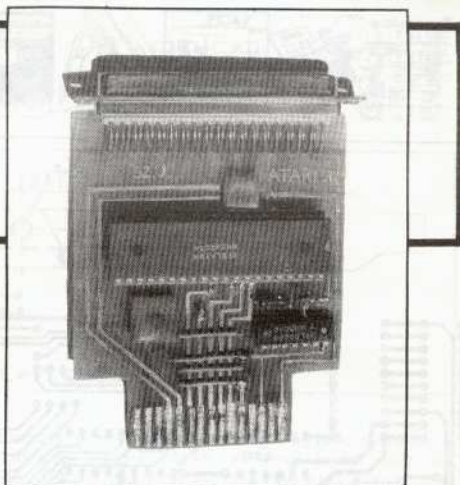
{ Pascal tesztprogram a
8 IN-8 OUT kártyához
Bináris érték olvasása a bemenetről }
USES Crt;
TYPE str8=string[8];
VAR
  ii:byte;
  bb:string[8];
BEGIN
  bb:='';
  for ii:=0 to 7 do
    if (V and (1 shl(7-ii)))>0
      then bb:=bb+'1'
      else bb:=bb+'0';
  Binary:=bb;
END;
PROCEDURE Toltas;
BEGIN
  ClrScr;
  GotoXY(35,10);
  writeln('A port értéke');
  repeat
    GotoXY(35,12);
    write(Binary[Port[$300]]);
  until KeyPressed;
END;
BEGIN
  Toltas; { Foprogram }
END.
  
```

```

{ Pascal tesztprogram
8 IN-8 OUT kártyához
Bináris érték írása a kimenetre }
USES Crt;
TYPE str8=string[8];
VAR
  sor:str8;
FUNCTION Decimal(sor:str8):byte;
VAR
  szam:
  ii:byte;
BEGIN
  szam:=0;
  for ii:=0 to 7 do
    if sor[ii+1]='1' then
      szam:=szam+1 shl(7-ii);
  Decimal:=szam;
END;
PROCEDURE Iras;
BEGIN
  writeln('Kérem a bináris számot');
  readln(sor);
  if length(sor)=8 then
    begin
      Fort[$300]:=Decimal(sor);
      writeln('OK.'):
    end;
END;
BEGIN
  Iras; { Foprogram }
END.
  
```

# Párhuzamos interfész

Az Atari 130XE, 65XE, 800XL és 800XE számítógépek mind egyikének van egy nyolcbites párhuzamos portja, amely a botkormányillesztőkre van kivezetve. Igényesebb illesztéseknél azonban nem elég a nyolc be/kimeneti bit és a két bemenő TRIG bit, ezért külön interfészre van szükség. Az itt leírt interfész a gép cartridge-csatlakozójához illeszthető. Mivel Magyarországon a Motorola 6520A PIA áramkörre nehezen szerzhető be, választásunk az Intel 8255A áramkörre esett.



## A kapcsolásról

Az elvi kapcsolási rajz az 1. ábrán látható. Az ábra bal oldalán a számítógép cartridge-csatlakozójáról levett jelek, jobb oldalán pedig az interfész kimenő csatlakozójának jelei láthatók.

A 8255-ös kiválasztása egyszerű, mert adott a bővítő áramkört engedélyező jel,  $\overline{CCTL}$ , amely a  $\$D500$ — $\$D5FF$  tartományban aktív. A teljes dekódolás tehát felesleges lenne, ezért a  $\overline{CCTL}$  jelet egyenesen a 8255 engedélyező bemenetére,  $\overline{CS}$ -re vezettük. A 8255-ös adatkivezetései, valamint  $A_0$  és  $A_1$  címkivezetései a gép megfelelő kivezetéseire csatlakoznak. A fent leírtakból következik, hogy a  $\$D500$ — $\$D5FF$  tartományon belül a 8255 regiszterei periodikusan ismétlődnek:

- 54528 =  $\$D500$ ... A port regisztere (PA)
- 54529 =  $\$D501$ ... B port regisztere (PB)
- 54530 =  $\$D502$ ... C port regisztere (PC)
- 54531 =  $\$D503$ ... parancsregiszter (CWR)
- 54532 =  $\$D504$ ... A port regisztere
- 54533 =  $\$D505$ ... B port regisztere

A 8255-ös RES bemenete tartósan L szintre van kapcsolva, ezért az interfész csak a gép bekapcsolásakor állítódik kezdeti állapotba (0-ás üzemmód, minden port bemenet).

A 8255-ös  $\overline{RD}$  és  $\overline{WR}$  jeleit a gép  $R/\overline{W}$  jeléből alakítottuk ki a 74LS00 IC kapuinak segítségével, melyek ilyen-

kor inverterként szolgálnak. Az R1 C2 taggal képzett impulzusrövidítő azért van szükség, mert írás műveletnél a 8255-ös  $\overline{WR}$  bemenő jelének inaktív válása után legalább 30 ns-ig kell még az adatsínen az adatoknak érvényesnek lenniük. A gép  $R/\overline{W}$  jele azonban éppen akkor változik L-ről H-ra, amikor a címsínen és az adatsínen lévő adatok érvényessége megszűnik.

Az  $R/\overline{W}$  jel írás műveletnél mintegy 560 ns-ig L szintű. Ezt az L szintű impulzust rövidíti az R1 C2 taggal képzett impulzusrövidítő hozzávetőlegesen 500 ns-ra, s így az időzítés a 8255 számára megfelelő. Olvasáskor ez a probléma nem áll fenn, ezért az  $\overline{RD}$  jelet az  $R/\overline{W}$  jel invertálásával hozhatjuk létre.

A  $\overline{WR}$  kialakításához kínálkozik egy

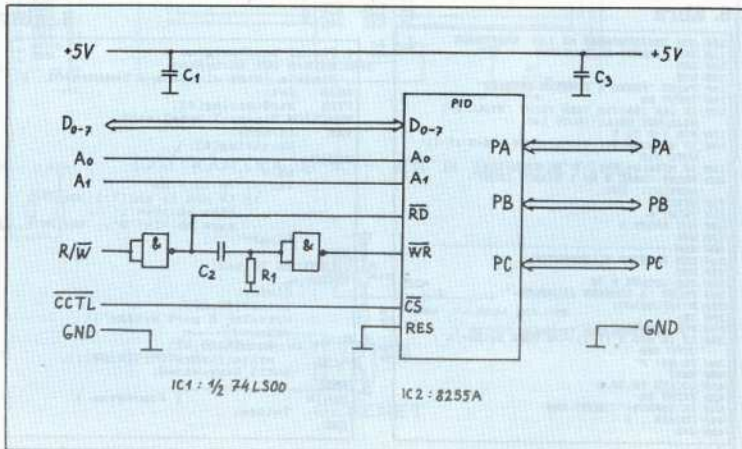
másik lehetőség is: az  $R/\overline{W}$  jel  $\Phi 2$  órajellel való kapuzása. Mivel azonban a  $\Phi 2$  félperiódusa mintegy 280 ns, ez a módszer nem tenné lehetővé a  $\overline{WR}$  jel szükséges hosszát (min. 400 ns).

## Az interfész elkészítése és élesztése

A kimeneti Canon-csatlakozó, valamint a számítógép cartridge-csatlakozójának bekötési rajza a 2. ábrán, a nyomtatási rajz a 3/a, 3/b ábrán, az alkatrész-beültetési rajza pedig a 4. ábrán látható. (A „..” a kétoldalalásírtá utal.) A NYÁK méretei  $63 \times 85$  mm, kivitelezése kétoldalas.

A NYÁK-ot kimaratása után erős fény felé tartva győződjünk meg ar-

1. ábra





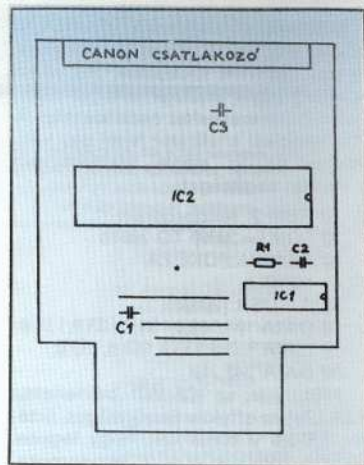
CANON CONNECTOR

A7	A5	A3	A1	NC	GND	NC	5V	NC	C7	C5	C0	C2	B0	B2	B4	B6	5V
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
A6	A4	A2	A0	NC	GND	NC	5V	NC	C6	C4	C1	C3	B1	B3	B5	B7	
																	37

NC...nincs csatlakozás

ATARI CARTRIDGE CONNECTOR

EN4	GND	A4	A5	A6	A7	A8	A9	A12	D3	D7	A11	A10	R/W	S2
A	B	C	D	E	F	H	J	K	L	M	N	P	R	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S4	A3	A2	A1	A0	D4	D5	D2	D1	D0	D6	S5	5V	EN5	CTL



4. ábra

2. ábra

ról, hogy nincs-e valamelyik oldalon zárlat vagy szakadás. Ha nincs hiba, hozzáláthatunk az alkatrészek beültetéséhez. Az alkatrészoldalon elhelyezett négy vízszintes átkötést szigetelt huzaldarabokkal készítsük el, az IC2 29. lábánál található átkötést pedig rövid huzaldarab beforsztatásával. A NYÁK azon pontjain, ahol az alkatrészoldalon is található forrasztószemek, a megfelelő alkatrészkievezetést mindkét oldalon meg kell forrasztani!

Ha elkészültünk a beültetéssel, megkezdhetjük az élesztést. Az interfész áramkörlapját úgy helyezzük a

cartridge-csatlakozóba, hogy a forrasztási oldal legyen felül, illetve az Atari 800XL-nél a felénk közelebb lévő oldal a forrasztási oldal legyen. Az interfész csatlakoztatása után kapcsoljuk be a számítógépet, és írjuk be, majd indítsuk el az alábbi rövid programot:

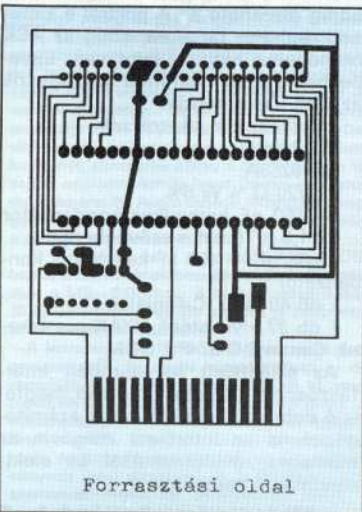
```

10 REM 1. tesztprogram
20 POKE 54531,128: POKE 54528,85
30 PRINT PEEK (54528)
    
```

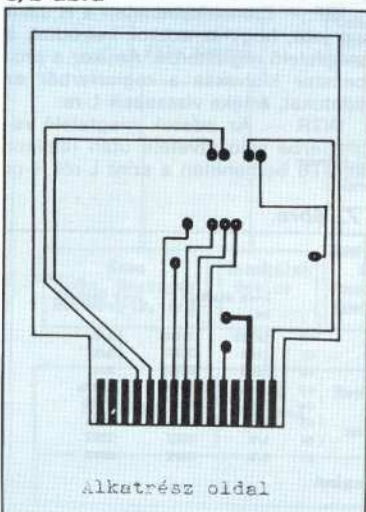
Ha a tesztprogram 85-öt ír ki, az élesztéssel elkészültünk, bár a teljes tesztelés ezzel még nem ért véget.

Ha a tesztprogramnak negatív az eredménye, kapcsoljuk ki a gépet, vegyük ki az interfészt és cseréljük ki a C2=1, 8 nF kondenzátort 1,5 nF-ra, majd ismételjük meg az előző tesztet. Ha ennek eredménye még most is negatív, próbálkozzunk C2=2,2 nF értékkel. (Megjegyzés: az R1 C2 tag funkciójáról lásd az áramkör ismertetéséről szóló fejezetben leírtakat.)

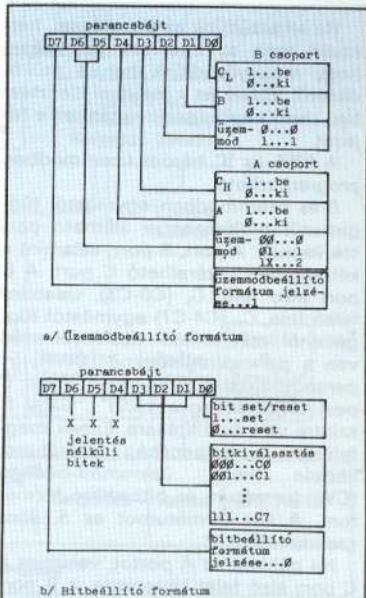
3/a ábra



3/b ábra



5. ábra



Az addig elkészített interfészek C2=1,5 nF és 1,8 nF értékű kondenzátorral egyaránt működtek. Ha valaki esetleg egyik értékkel sem jár sikerrel, a hibakereséshez oscilloszkóp, illetve logikai analízátor híján egy voltméter és az alábbi tesztprogram nyújthat segítséget:

```
10 REM 2. tesztprogram
20 FOR I=20000 TO 20018
30 READ A:POKE I,A
40 NEXT I
50 A=USR(20000)
60 DATA 104,169,0,133,0,133,0,133,0
70 DATA 133,0,133,0,133,0,133,0
80 DATA 240,240
```

Mérjük le az IC2  $\overline{WR}$  bemenetén (36. láb) az effektív feszültséget, értéke 3,5-3,6 V körüli kell, hogy legyen. Ezután indítsuk el a 2. tesztprogramot, és újra mérjük le a  $\overline{WR}$  bemeneten az effektív feszültséget. Értéke most 3,2-3,3 V körüli kell, hogy legyen. A 2. tesztprogramból csak RESET-tel lehet kilépni. Ha a 2. tesztprogram negatív eredményt hozott, a hiba az R/W jel átalakítása körül keresendő (R1, IC1, NYÁK). Pozitív eredménynél az IC2 körül, esetleg magában az IC2-ben lehet a hiba. Ha a NYÁK-ot figyelmesen készítettük el és jó minőségű alkatrészeket használtunk, az élesztés nem okoz komoly gondot.

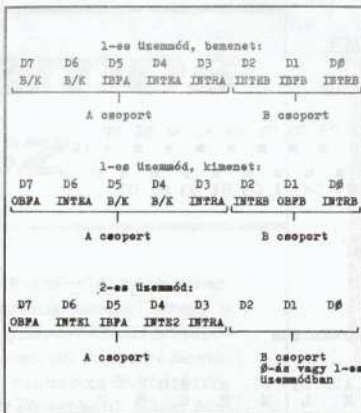
## Az interfész számítógépes vezérlése

Ha elkészültünk az élesztéssel, hátravan még az interfész tesztelése, hogy meggyőződjünk helyes működéséről. Ehhez és a majdani illesztéshez kíván segítséget nyújtani ez a fejezet.

A 8255-ös IC három üzemmódban programozható.

0-ás üzemmódban egymástól függetlenül be/kimenetre állítható portok vannak: A port, B port, valamint a két részben vezérelhető C port. A C port also fele,  $C_L$  (C0-C3), valamint felső fele,  $C_H$  (C4-C7) egymástól függetlenül vezérelhető. Ezenkívül mód van a parancsregiszterbe (CWR) irt parancsbájttal a kimenetre állított C port bitjeinek egyenkénti L vagy H szintre való belállítására. Ennek megfelelően a parancsbajt formátuma kétféle lehet: üzemmód-beállító (CW) formátum és bitbeállító formátum. A két formátumot az 5. ábra szemlélteti.

Ha például az A portot, valamint a C port alsó felét kimenetre, a B portot, valamint a C port felső felét be-



6. ábra.

menetre akarjuk állítani 0-ás üzemmódban, akkor a megfelelő parancsbajt binárisan 10001010, azaz tízes számrendszerben 138. Esetünkben tehát POKE 54531,138.

Az 1-es üzemmódban egyirányú handshaking (kézfogásos) adatátvitel valósítható meg. Adatátvitelre az A és/vagy a B port használható, irányításra a C port. A C portot olvasva 1-es és 2-es üzemmódban az állapotbajtot kapjuk, melynek formátumai az egyes üzemmódokban a 6. ábráról olvashatók le. Az irányító jeleknek a C port egyes bitjeihez való hozzárendelése a 7. ábrán látható.

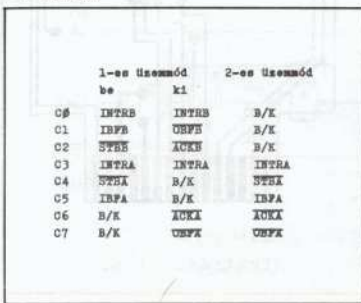
Bemenetnél az irányító jelek funkciója 1-es és 2-es üzemmódban a következő:

**STB** — Ha ezen a bemeneten L szint van, az adatok beíródnak a megfelelő port regiszterébe.

**IBF** — Ezen a kimeneten a H szint azt jelzi, hogy az adatok beíródtak a megfelelő regiszterbe. Amikor a processzor kiolvassa a regiszterből az adatokat, értéke visszaesik L-re.

**INTR** — Az adatok megfelelő regiszterbe való átvétele után (amikor az STB bemeneten a szint L-ről H-ra

7. ábra.



változik) az INTR kimenet H-ra állítódik be. Az adatok regiszterből való kiolvasása után — amikor a processzor kiolvassa a regiszterből az adatokat — értéke visszaesik L-re.

INTR jel azonban csak akkor keletkezik, ha a megfelelő INTEA, illetve INTEB bit a C port regiszterében 1-re van állítva (a parancsbajt beállító formátumának felhasználásával). A 2-es üzemmódban a bemenetkor keletkező INTR jelet az INTE2 bit engedélyezi (6. ábra).

Kimenetnél az irányító jelek funkciója 1-es és 2-es üzemmódban a következő:

**ÖBF** — Ha ez a jel L szintre esik, azt jelzi, hogy a processzor beírta az adott port regiszterébe a kívánt adatokat. A periférikus egység ACK jelének aktívá válása után az ÖBF visszamegy H szintre.

**ACK** — Ezen a bemeneten az L szint azt jelzi, hogy a periférikus egység átvette az adatokat az A, illetve a B portról.

INTR — Akkor lesz értéke H, ha  $\overline{ACK}=H$ ,  $\overline{ÖBF}=H$  és a megfelelő INTE bit 1-re van állítva. L-re akkor esik le az INTR kimenet, amikor a processzor beír az adott port regiszterébe. A 2-es üzemmódban a kimenetkor keletkező INTR=H jelet az INTE1 bit engedélyezi.

A 2-es üzemmódban kétirányú kézfogásos adatátvitelre van mód az A porton keresztül. Az A port kivezetési ebben az üzemmódban nyolcbites kétirányú adatszátonának felelnek meg, be- és kimeneti regiszterekkel. Az irányítást a fent említett jelek végzik. Az A port kivezetési magas impedancia állapotában vannak. Csak addig olvasható az A portról a kimeneti regiszter tartalma, amíg az ACK bemeneten aktív L szint van. Egyebekben a 2-es üzemmód megegyezik az 1-es üzemmóddal.

*A felhasznált alkatrészek*

IC1 74LS00  
IC2 8255A  
R1 220  $\Omega$  1/4 W 5%  
C1 150 nF keramikus kondenzátor  
C2 1,8 nF (lásd a szövegben)  
C3 68 nF — 100 nF keramikus kondenzátor

1 db 40 lábú IC-foglalat  
1 db 37 kivezetésű NYÁK-csatlakozó, Canon DC 37 P 1 BON

Az elkészített és kipróbált interfészpanel ajánlatos mielőbb megfelelő dobozba helyezni, ami a számítógépet és az interfészt megóvja az esetleges rövidzárlatoktól és elektrosztatikus sokkaktól.

Máder Indira—Szijártó Jenő

# Az AmigaDOS

sit. Megjegyzés: ha a fájl nevében szerepelnek a fenti speciális karakterek, akkor a fájlnevet aposztróf (') jelek közé kell tenni.

A rendszerlemezről található könyvtárak a következőket tartalmazzák:

SYS	a rendszerlemez gyökérkönyvtára;
C	az AmigaDOS utasításait tartalmazó könyvtár;
L	ez a könyvtár a legerjedelmesebb utasítások szegmenseit és az operációs rendszer tranzienzi részeit tartalmazza;
S	itt található az a Startup-Sequence nevű — AmigaDOS-ban írott — fájl, amely a rendszer indulásakor végrehajtható. (A Startup-Sequence MS-DOS megfelelője az AUTOEXEC.BAT.) A könyvtár különleges tulajdonsága, hogy ha az Execute parancs nem találja a gyökérkönyvtárban a végrehajtandó fájlt, akkor itt fogja keresni;
LIBS	a gépi kódú könyvtárak megnyitására szükséges fájlokat tartalmazza;
DEVS	az egyes készülékek kezeléséhez szükséges rutinokat találjuk meg benne. Itt található a Mountlist (lásd: Mount utasítás) és a System-Configuration. Ez utóbbi

Az AmigaDOS — mint a Unix leszármazottja — az Amiga multitasking szervezésű operációs rendszere. A multitasking azt jelenti, hogy egy számítógépen egyidejűleg több programfeladat (task) futhat. Minden tasknak saját prioritása, elsőbbségi szintje van. Az egyes taskok külön ablakokat használnak a képernyőre való kivitelhez, és felosztják egymás között a memóriát. Minél több task fut a gépen, annál lassabb a működésük az időosztásos (time-sharing) üzemmód miatt. A multitasking miatt az Amigát alacsony szintű nyelven — C-ben vagy assemblyben — programozni meglehetősen bonyolult feladat. Szabályok sorát kell betartani ahhoz, hogy a különböző programok ne zavarják egymást. Jó segítséget nyújt ehhez a Kickstart-ROM könyvtárainak használata. A számítógépben állandóan külön task fut az RS232-és port, a billentyűzet, külön-külön minden lemezmeghajtó és a CLI (Command Line Interface, az AmigaDOS parancsértelmezője) kiszolgálására.

Az Amiga rendszerszoftvere modulokból épül fel. Ezek egy része a Kickstart-ROM-ban helyezkedik el, több pedig szükség szerint lemezzel tölthető be. Az 1. ábra az egyes modulok kapcsolatát szemlélteti.

A rendszerszoftver legmagasabb szintje a Workbench és a Command Line Interface (CLI), amelyek a felhasználó által is „láthatók”. A Workbench az Intuition nevű modul a képernyőkezeléshez, az AmigaDOS-t pedig a fájlkezeléshez használja. Az Intuition viszont az Input Device-szel a bemeneteket, a Graphics és a Layers könyvtárakkal pedig a kimeneteket éri el.

Az AmigaDOS tartja kézben a fájlkezelést, önmaga pedig az Execre épül, amely a taskokért, a megszakításokért, a rendszerüzemeltetés továbbításáért, a B/K-ért és sok más funkcióért felelős. A Trackdisk Device a legalacsonyabb szintű lemezkezelő interfész, ami az olvasófej mozgatóját és az írási/olvasási műveletet látja el. A Keyboard és a Gameport Device kezeli a billentyűzetet, az egeret és a botkormányt, sorrendbe állítja a bemeneteken lezajlott eseményeket az Input Device nevű modul számára. Az Audio Device feladata a hanggenerálás, a Serial és a Parallel Device-é pedig a portok közvetlen kezelése.

Az utasítások és a programok AmigaDOS-ban is a fájl nevének beadása útján indíthatók, akár a MS-DOS-ban. A fájlnevé maximum 30 karakterből állhat. Ha szóközt is tartalmaz, akkor idézőjelbe kell tenni.

A lemezen lévő fájlok a könyvtárakkal csoportosíthatók. Az egyes könyvtárakban — directorykban — fájlokat helyezhetünk el, vagy újabb könyvtárakat is. Ezeket alkönyvtáraknak — subdirectoryknak — nevezik. A lemez legelső, alapkönyvtára, amelyre a lemez vagy a meghajtó nevével hivatkozhatunk, az ún. gyökérkönyvtár (root directory). A 2. ábra egy rendszerlemezről mutat be, amelyen jól látszik a könyvtárstruktúra jellegzetes fa szerkezetű fel-

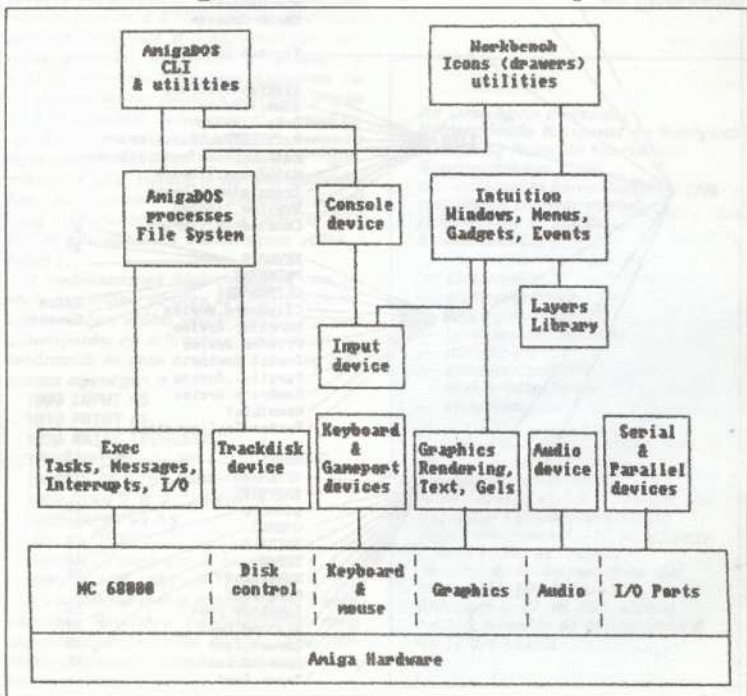
építése. A fa egyes ágain különböző fájlokhoz juthatunk el. Az elérés közben érintett könyvtárak — az elágazások — sorát útvonalnak, pathnak hívják. A gyökérhez eggyel közelebb levő könyvtár neve angolul parent (szülő).

Ha ebben a rendszerben hivatkozni akarunk egy fájlra, akkor annak a neve előtt / jellel — törvonallal — elválasztva fel kell sorolnunk az érintett könyvtárakat, vagyis fel kell tüntetnünk az útvonalat. Az útvonalnak az aktuális könyvtárból kell kiindulnia. (Az aktuális könyvtár megadásához lásd a CD parancsot.)

Ez a DOS is lehetőséget nyújt a fájloknak rövidítésére, mégpedig az alábbi kód szerint:

- ? egy tetszőleges karaktert helyettesít,
- % üres sztringet jelent,
- \* karakter a megadott karaktert vagy üres sztringet jelent,
- {kar1, | kar2} a két karakter valamelyikét helyettesíti,
- {kar1kar2kar3...} a karakter sor egészét helyettesítheti,
- \*7, egy tetszőleges karakterláncot helyettesít.

## 1. ábra. Az Amiga rendszerszoftverének felépítése



a rendszer főbb működési paramétereit tartalmazza, és a Preferencés nevű programmal módosítható;

**FONTS**  
**T**  
 a lemeztől betöltődő karaktereket tartalmazó könyvtár;  
 egyes programok ideiglenes munkafájlok tárolására használják ezt a könyvtárat. Például az Ed nevű szövegszerkesztő itt helyezi el a módosított szöveg eredetijét „-backup” kiterjesztéssel.

Az egyes készülékek, így a lemezmeghajtók neveit is kettősponttal kell lezárni. Az AmigaDOS az alábbi készülékeket különbözteti meg: DF0...DF3 négy különböző hajlékonylemez-meghajtó, melyek közül a DF0 a beépített;

DH0, DH1 két winchester-meghajtó;  
 RAM RAM-lemez (a RAM-ot mint háttértárat kezeli a rendszer);  
 NIL virtuális — látszólagos — meghajtó;  
 SER soros port (RS232);  
 PAR párhuzamos port (Centronics);  
 PRT nyomtató;  
 CON képernyő;  
 RAW a képernyő mint soros be- és kiviteli eszköz.

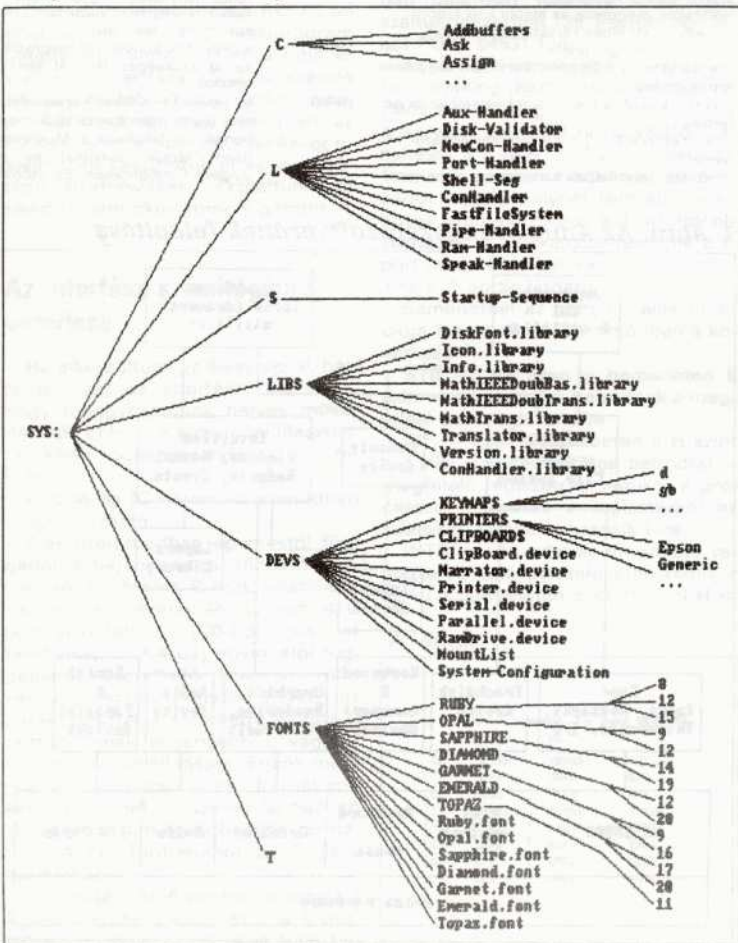
Az Assign paranccsal egyébként tetszőleges könyvtárakat is hozzárendelhetünk az egyes készülékekhez, és ezután ezekre is, mint készülékekre hivatkozhatunk. Egy ilyen hozzárendelést a rendszer betöltésekor automatikusan elvégez a DOS, ezért a rendszerlemez SYS: néven is elérhetjük.

A továbbiakban az AmigaDOS 1.2 verziójának utasításkészletét ismertetem, amelynek során a következő jelölési rendszert használom:

- a DOS utasítások és a hozzájuk tartozó, szintén a DOS részét alkotó kiegészítések, opciók stb. nagybetűvel írottak,
- a kisbetűs szavak helyére a felhasználónak kell behelyettesítenie a konkrét paramétereket,
- a szögletes zárójelek között álló részek megadása tetszőleges, az utasítás nélkülük is működik,
- a törtvonallal elválasztott részek azt jelentik, hogy a felhasználónak választania kell a felsorolt lehetőségek közül.

Az utasításkészlet után a hibáüzenetekről is összefoglaló listát adok közre.

## 2. ábra. A rendszerlemezén elhelyezkedő rendszerkönyvtárak struktúrája



## Az AmigaDOS utasításkészletének összefoglalása

### [utasítás]; [megjegyzés]

Az utasítássorok végére pontosvesszővel elválasztva írhatunk megjegyzéseket.

### utasítás[> ]/[<][argumentumok]

A kisebb/nagyobb jel nyílként szolgál, és az adatrányt mutatja. A „LIST>Tartalom” utasítás hatására például a DOS a lemez tartalomjegyzékét mint szöveges fájlt elmenti Tartalom néven.

### ADDBUFFERS [DRIVE] meghajtó [BUFFERS] n

A meghajtó által használt puffer méretét n értékkel megnöveli. A puffer megnövelése gyorsabbá teszi az írás/olvasás műveletet. Az n=20 beállítása mintegy 10 kb-átos puffert hoz létre. Az n=25-nél nagyobb érték már nem növeli tovább a sebességet, mert ekkor egy írás/olvasás hozzáférés során egy teljes sávot ír/olvas a számítógép.

### ASK [kérdés]

Kírja a kérdést és Y vagy N (igen vagy nem) beadására vár. Y esetén 20-as hibakód jön létre, vagyis WARN igaz lesz (lásd IF utasítás), N esetén pedig a hibakód 0 marad.

### ASSIGN [[régí logikai név] új logikai név][LIST]

Az eddig a régi logikai néven nyilvántartott lemezre ezután az új logikai néven is hivatkozhatunk. LIST kiterjesztésnél kilistázza a fennálló összes logikai hozzárendelést. Figyelem! Az ASSIGN nem azonos a RELABEL paranccsal, amely fizikailag is megváltoztatja a lemez nevet, ez csak egy logikai művelet!

### BINDDRIVERS

Sorba rendezi az AmigaDOS számára a rendelkezésre álló meghajtókat. (Ezt az utasítást a Startup-Sequence-ben célszerű elhelyezni.)

### BREAK takszám [ALL] [C] [D] [E] [F]

Beállítja, hogy a megadott számú taskban mivel lehessen az éppen futó programot megállítani. Beállítható CTRL+C, CTRL+D, CTRL+E, CTRL+F vagy ALL opcióval mind a négy billentyűkombináció. Az alapértelmezés: CTRL+C, a rendszer betöltésekor pedig CTRL+D.

### CD útvonal

Az aktuális könyvtárakat a megadott útvonallal módosítja. Az egyes könyvtárak neveit

# Adom a magyarázatot!

törtvonalal kell elválasztani. A gyökérványtárba kettőspont beadásával, a gyökérhez egygel közelebbi könyvtárba (parentbe) törtvonalal jutunk.

## CHANGETASKPRI [PRI] prioritás

A task prioritását a megadott értékére állítja be, amely -128 és +128 közé eshet. Alaphelyzetben minden task prioritása 0.

## COPY [[FROM] forrás fájlnev] [TO cél fájlnev] [ALL] [QUIET]

A forrásfájl tartalmát átmásolja a célfájlba. ALL esetén a könyvtár minden elemét - és az alkönyvtárak minden elemét is átmásolja. QUIET kiterjesztésekor nem írja ki a lemásolt fájlok neveit.

## DATE [nap-hónap-év] [óra: perc] [TO/VER fájlnev]

Paraméter nélküli esetben kiírja a rendszeridőt. A paramétereket megadva beállítja a rendszeridőt, és ha a fájlnevet is megadjuk, akkor annak létrehozási időpontját módosítja.

## DELETE [fájlnev... ] [ALL] [Q/QUIET]

Fájlokat töröl, melyek száma maximálisan tíz lehet. ALL esetén a könyvtár és az alkönyvtárak minden elemét töröl. QUIET megadásakor mindezt visszajelzések nélkül végzi.

## DIR [útvonala] [OPT A/I/Al]

Kiírja a megadott útvonalon levő vagy az aktuális tartalomjegyzéket. OPT A esetén minden útvonalat kilistáz. OPT I-t megadva interaktív módon hajtja végre a parancsot. Ekkor az alábbi billentyűfunkciók élnek:

RETURN = újabb sor kiírása

Q = kilépés

B = egygel visszalép az aktuális könyvtárból, a parentbe

E = könyvtárnév esetén belép az illető könyvtárba

T = karakteres formában kiírja a fájl tartalmát CTRL + C = kilépés, mint általában

## DISKCHANGE [DEV] meghajtó

Ezzel a paranccsal közzölhetjük a rendszerrel, hogy az adott meghajtó egy 5,25"-os lemezegység.

## DISKCOPY [FROM] forrás lemeznév TO cél lemeznév NAME nev

A forráslemez tartalmát teljes egészében átmásolja a céllemezre a megadott új néven.

## DISKDOCTOR [DRIVE] meghajtó

Diagnózist készít a meghajtóban lévő lemeztől. Hibás lemezek rendezésére, tisztázására használatos utasítás. Tételeken felsorolja a hibás fájlokat, sávokat és szektorokat. Ha a parancs végrehajtása befejeződött, ajánlatos a lemez tartalmát fájlönként átmásolni egy hibátlan lemezre, és a hibás lemezt újraformázni.

## ECHO szöveg

A megadott szöveget kiírja a képernyőre. Ha a szöveg szöveget is tartalmaz, akkor idézőjelek közé kell tenni. Ha ASCII vezérlőkéretek után írott betű formájában adhatók meg. Például: "N=new line = új sor vezérlőkéretek."

## ED [FROM] fájlnev [SIZE munkaterület]

Behívja az AmigaDOS szövegszerkesztőjét, és szerkesztésre betölti a megadott nevű szöveges fájlt, amelynek lefoglalja a kívánt méretű munkaterületet. Ha ezt nem adjuk meg, az alapértelmezés 4000 bjt.

## ENDCLI

Befejezi azt a CLI taskot, amelyben az utasítást kiadtuk, és bezárja a számára fenntartott ablakot. (Folytatjuk)

*A kérdés az 1989/5. számunkban jelent meg, ezért röviden megismétljük. Szekvenciális állományt olvastunk lemezről, és olvasás közben lemeredvezt a Commodore úgy, hogy még a RUN/STOP is háttástan maradt. Megállapítottuk, hogy a gép éppen egy INPUT\* utasítás végrehajtása közben vált működésképtelenné.*

*A feladványban szereplő szekvenciális állomány hossza a lemezen nulla volt, erre a feladványból bárki rájöhetett. Az okokra azért volt nehéz rájönnöm, mert a szekvenciális állomány napokkal korábban hoztam létre, és elfelejtettem már, hogy milyen körülmények között. Nos, mi történt hát?*

*A készítés alatt álló program billentyűzet-INPUT üzemmódban volt, de (még) zavaros képernyőkép miatt nem írtam be adatot, hanem a STOP billentyűvel megállítottam a gépet. A megnyitott szekvenciális állomány megjelent a lemezen, a LED égve maradt, de nem figyeltem fel rá. Óvatos ember lévén, a program elején CLOSE utasítást helyeztem el, s ezért újraindítások a program lezárta a lemezes egységen korábban megnyitott állományt. A LED kialudt, a tartalomjegyzékben szabályosan lezárt állomány jelent meg. (Ellenkező esetben a WRITE FILE OPEN hibajelzésből észrevettem volna a hibát.)*

*A védekezésnek több módja is van. Az első: megelőzni az üres állomány létrehozását. Ehhez előbb végrehajtunk minden billentyűzési és aritmetikai műveletet, ellenőrizzük, és csak aztán nyitjuk meg a lemezes egységen a fájlt:*

```
1000 INPUT AS
1010 PRINT AS
1020 PRINT "RENDBEN?"
1030 GET BS: IF BS="" THEN 1030
1040 IF BS="N" THEN 1000
1050 OPEN 2, 8, 2, "MINTA, S, A"
1060 PRINT #2, AS
1070 CLOSE2
Beírhatjuk még azt is, hogy
1045 IF LEN (AS) = 0 THEN STOP
```

*A védekezés másik módja, hogy a szekvenciális állomány beolvasásánál megakadályozzuk a nulla hosszúságú állomány olvasását. Ehhez próbásképpen meg kell nyitnunk:*

```
2000 OPEN 2, 8, 2, "MINTA, S, R"
2010 GET #2, AS: CLOSE2
2020 IF AS="" THEN STOP
2030 IF AS = CHR$(13) THEN STOP
2040 REM MINDEN RENDBEN
2050 OPEN 2, 8, 2, "MINTA, S, R"
2060 INPUT #2, AS
2070 CLOSE2
```

*A 2010-es sorban azért áll GET\* utasítás, mert azt a Commodore végrehajtja, nem fogja le, még akkor sem, ha a beolvasott AS üres. A fent említett védelmet a 2020-as sor látja el. A 2030-as sorban járulékos védelem áll arra az esetre, ha a példában említettéhez hasonló módon - hibba folytán - a lemezre csupán egy RETURN karakter került volna. (Ez is előfordult már.)*

Dr. Zana János

**Az Országos Műszaki Információs Központ és Könyvtár (OMIKK) Referáló Kiadványok Szerkesztősége külső munkatársakat keres külföldi folyóiratok magyar nyelvű referálására az alábbi szakterületeken:**

- elektronika
- elektrotechnika
- fizika (alkalmazott)
- gyártásautomatizálás
- mélyépités
- számítástechnika
- távközléstechnika
- vízellátás

*Elsősorban angol, német, francia és orosznyelv- ismerettel rendelkező szakemberek jelentkezését várjuk a 181-994, ill. 382-300/113 telefonszámokon (munkanapokon 8-17 óráig) vagy személyesen az OMIKK anyaggyűjtési és nyilvántartási csoportnál (Budapest VIII., Múzeum u. 17. III. 301. szoba) hétfőn, szerdán és csütörtökön 8 és 17 óra között.*

# Figuratípusok jutalompontjai

A sakkjáték fontos tényezője, hogy a király hol helyezkedik el a táblán és az, hogy milyen figurák helyezkednek el körülötte. A közepen rekedt királyra az ellenfél könnyen csapást mérhet, mert miután a megnyitásban a centrumgyalogok elmozdultak a helyükről, a közepén maradó királyt gyalogok nem védik. Ezáltal a közepen megnyíló vonalakon, átlókon keresztül hamar az ellenséges figurák célpontjává válik. Ezért a sakkjátékban szinte nélkülözhetetlen a sáncolás.

A sakkjátékosok arra törekszenek, hogy királyukat mielőbb biztonságba helyezték. Ezért először a király és a bástya közötti könnyűszi-  
teket szokták kifejleszteni, hogy azok ne akadályozzák a sáncolást. Akadnak kivételek is, amikor a sáncolás előtt hasznosabb lépéseket is tehetünk. Vannak megnyitási változatok, amelyekben csak a tizenötödik és a huszadik lépés között kerül sor a sáncolásra. A sáncolás lehetőségét azonban — még ha nem is élünk vele azonnal — célszerű mindig készenlétben tartani.

A játszmák többségében rövid sáncot alkalmaznak, ugyanis ehhez csak két bábuval, a húszárral és a futóval kell lépni, míg hosszú sánc esetén a vezérnek is el kell hagynia alapállásbeli helyzetét. Ez utóbí viszont nem szerencsés, mert a vezér a könnyűszi-  
teket célpontjává válhat, és ezért tempóvesztésekkel kell visszavonulnia. A rövid sáncot más szempontból is biztonságosabbnak tartják a hosszú sáncolásnál. Rövid sánc esetében a királyt f2, g2, h2 gyalogok védik, míg hosszú sánc esetén, amikor a király c1-re kerül, e feladatot a d2, c2, b2 gyalogok látják el. A vezérfutó lépése előtt a d és a b gyalogok közül legalább az egyikkel el kell lépünk, hogy lehetővé váljék a hosszú sáncolás. A királyt védő gyalogok közül legalább az egyiknek el kell lépnie a helyéről, s így a királyállás eleve meggyengült. Ezenkívül védtelen marad az a-vonal, az a2 (a7) gyalog hosszú sánc esetén pedig újabb támadható pontot jelent az ellenfélnek. Ezért a királyállás biztonságosabbá tétele érdekében szükség van még a Kc1-b1 lépésre is, ami újabb tempóvesztést jelent.

A király csak a végjátékban indulhat meg biztonságosan a centrum felé, a centrummezőket elfoglalni, ellenkező esetben könnyen matot kaphat, ami a játszama végét jelenti. A megnyitásban és a középjátékban főként a

gyalogok és a könnyűszi-  
teket védik a királyt, a végjátékban viszont fordul a kocka; legtöbbször az előre tolt gyalogok — a tisztjelöltek — szorulnak a király védelmére. Ezért a játszma utolsó fázisában — a végjátékban — megnövekszik a király aktivitása; védi a saját gyalogjait és támogatja az ellenséges figurát.

A király aktivitását jól jellemezhetjük a mozgékony-  
sági értékkel, amely esetünkben nem más, mint a legális lépések száma (vö. az előb-

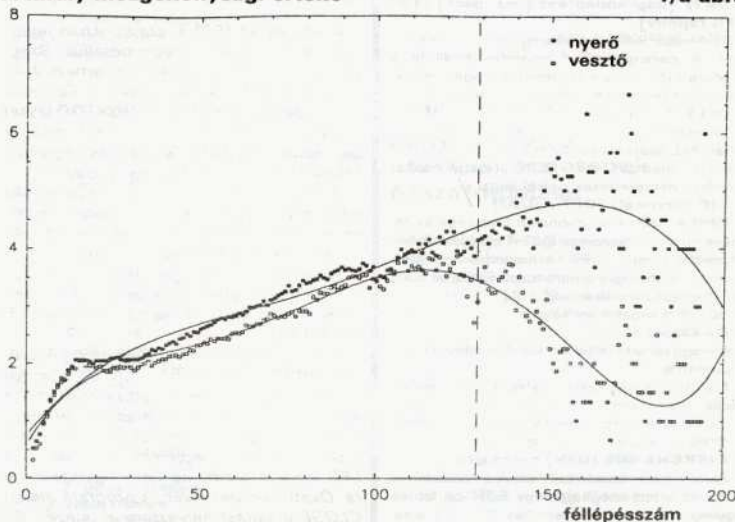
biekkel, ahol a mozgékony-  
ságot, mint a psze-  
dólegális lépések számát definiáltuk). A bemutatott hadállásban a világos király mozgékony-  
sági értéke három, a sötété négy.

Az 1/a ábrán a nyerő és a vesztő fél királyának mozgékony-  
sági értékét láthatjuk a féllépésszám függvényében, míg az 1/b ábra a nyertes és a vesztes mozgékony-  
sági értékének a különbségét mutatja a király esetében.

Kovács P. Attila

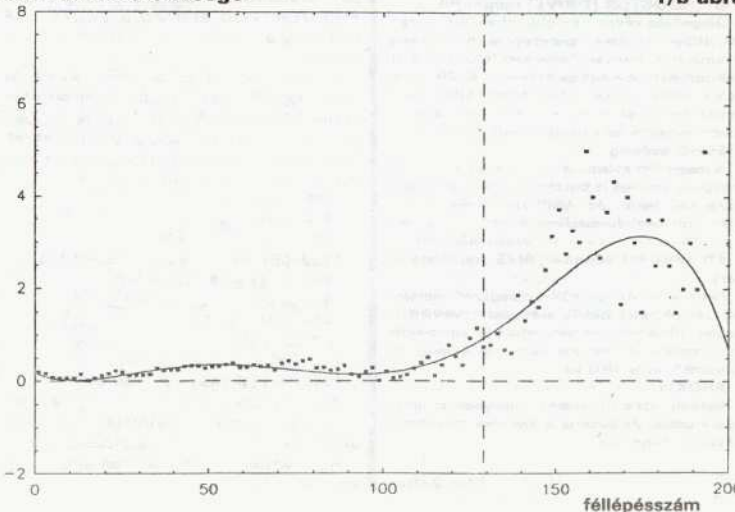
A király mozgékony-  
sági értéke

1/a ábra

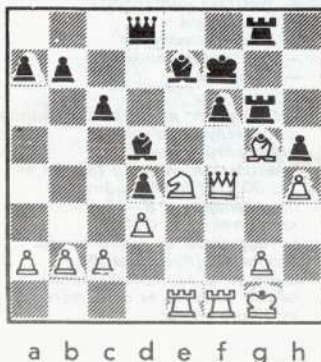


A nyerő és a vesztő király mozgékony-  
sági értékének különbsége

1/b ábra



Kasparov—Roizman, Minszk, 1978. 42. féllépés után létrejött hadállás



# PROGRAMTERMÉK

## Vegyészeti csemegék

A kémia kedvenc tantárgyaim közé tartozott. Rádásul az egyetlenem fizikus diákként is jó sokat tanítottak nekünk belőle. Időnként olyan érzésünk volt, hogy jóval többet kaptunk, mint amennyi fizikával végzés hallgatótársainkat „büntették”. Lehet persze, hogy ők meg éppen fordítva látták a helyzetet.

Mint a bevezetőből érezhető, most ismét valami kémia tárgy program kerül a terítékre. Nem is egy, hanem kettő. Szerzőik azonban azonosak. A Tudományszervezési és Informatikai Intézetben jártamban valami Plus/4-es program után kérdezősködtem, mert az utóbbi időben kevés ilyen programot böncogattam. Így bukkantam az összefoglaló adatok között olvasható szerzőpáros vegyészeti, pontosabban fizikai-kémiai tárgyú programjára.

A fizikai kémiát sokan elég száraz tantárgynak tartják. Ezért nem árt neki egy kis népszerűsítés. A két vizsgált program kiválasztott téma szerencsére önmagában is elég érdekes a figyelemfelkeltéshez. Ha ezt sikerül valami frappáns képi megjelenítéssel és animációval megtámasztani, akkor egészen émeszhető produktumok juthatnak.

A szerzőpáros eleget tett a kívánalmaknak. A képi megjelenítés szépségére való törekvés mindjárt a címeképen látható. S a programbel-sők ezt a stílust folytatják. Elég sok energiát öltettek ebbe a „szépészetbe”, de megérté. Alapelvek érhetően az, hogy ami nem szép, az nem is lehet jó. A két programra minden-esetre feltétlenül rámondható, hogy szép.

Nézzük a beltartalmakat. Hiába szép a küllem, ha netán a belsejével valami baj van. Esztétikusság so sincs ilyen gondról. A programok értéke összhangban van küllemükkel. Minde-nekelőtt a pontos helyesírás nyugtázott le. Számítógépek, írógépek, fertőző környezet-tünnben olykor meglehetősen bajba kerülünk a hosszú i, ó, ő, ú és ú írásával. Az emlegetett eszközök ugyanis durván eltorzítják az iskolá-ban hajdan megszerzett ismereteinket, ránk ló-szólvé korlátozott karakterkészleteiket. Most, miután egy-két évtizednyi (kinek-kinek mennyi) szünet után megnyílt a lehetőség a teljes ma-gyar ékezetes betűkészlet használatára, időnként alig emlékezünk, hogyan is kell egyik-má-sik szót írni. Szakmai ártalomtól szenvedünk. Bevallom ösztönten, hogy szótár után kaptam, amikor elolvastam a programban a „kísérlet” szót. Nyomasztó kísérés fogott el, hogy itt va-lami zúr van a hosszú i-vel. Nem! A szerzőknek volt igazuk, mint több más szónál is, melyek közül többnek is utánanéztém.

Mindenesetre nagyon jó, hogy a Plus/4 me-jelenítési a teljes magyar ékezetes betűkész-letet. Persze, ha használják is. És nemcsak ma-gyar irodalom és nyelvi témájú programokban!

Az új nemzedéknek látszólag nem kell már olyan nyelvhelyességi ártalmakat szenvednie a gépektől, mint nekünk. A baj csak az, hogy ne-hezen alakul az egységesség az alkalmazás módjáról (lásd Számítástechnika — Computer-world fórum a szabvány PC karakterkészlet ki-alakításáról, amellyel szemben mások a nem tökéletes logikájú — és a megvalósítás realitá-saitól messze elrugaszkodó, de sajnos még mindig érvényben lévő — MSZ szabvány sz-erinti készletet emlegetik napjainkban is e lap hasábjain, meg sem említve más, racionál-iabb és praktikusabb törekvéseket!). Ez a gon-dotlatmenet azonban túl messzire vezetne.

A megvizsgált programok futásideje nem túl hosszú ahhoz, hogy bőven beférjenek egy nor-mál tanórába. Előbb azonban túl kell jutni a ka-zetta olvashatóságával kapcsolatos zűrókon. Úgy érzem, nem árt ismételtten felhívni a fig-yelmet a Novotrade beállított kettazettájának hasznosságára. Továbbá a magnófejek kopá-sára. Egy-két évi használat után már elkelle a szervizelés. Nem beszélve a fejek alkohollal vá-ló rendszeres tisztogatásáról. A betöltési ne-hézségek komolyan akadályozhatják még ilyen rövid lélegzetű programok alkalmazását is a tanórákon, ha nincs mód a programok óra elő-tti betöltésére. A szakkrői foglalkozások levegő-jét is megmérgezi az olvasási hibák okozta ku-darcélmény.

A programok közül először a kémiai reakciók sebességét tárgyalót próbáltam ki. A program-mal kísérletek végezhetünk „szárazon”, ami kihasználja a számítógép fantáziát nem korlá-tozó hasznos tulajdonságát. Előny, hogy köz-ben a sósav még véletlenül sem csipi a tanuló szemét. Az így „végrehajtható” öt kísérletben megismerkedhetünk azzal, hogyan befolyásol-ja a kémiai reakciók sebességét az anyag mi-nősége, a részecskeméret, a belső energiaáll-a-pot, illetve a termikus hatás, a fényhatás és a katalizátorhatás. A katalizátorok hatásának elemzését kifejezetten élvezettel figyeltem. Azt hiszem, másoknak is tetszeni fog, és nem csu-pán az én „hozzaértéstől fertőző” tudatom el-foglaltsága mondatja velem ezt a véleményem.

A betöltéssel kapcsolatos előző fejtegetése-i-met a második program betöltésekor szerzett kudarcélmények alapozták meg. A türelem szép és jó programot terem. A fázadozás tehát megérté. Mindjárt látható volt az is, hogy vá-lószínűleg ez a program készült később. Továb-fejlesztették az előző szimpatikus vonásait, egy, az alsó sorban futó üzenetsorral, amely az aktuális teendők folyamatosan közli (help). Sajnos az öröme némi öröm is vegyült. A futó üzenetsor nagyon elfoglalja a processzort, ezért a billentyűzet reakcióképessége erősen degradálódott.

A korsztály tulajdonságainak megfelelően virgóc tanuló bízatosan nem értékeli majd, hogy a gépet hidegen hagyja, ha a választ csak egy pöccintéssel akarják elintézni. Jól meg kell „taposni” a billentyűket, hogy a program észre is vegye a választunkat. Emiatt menet közben karakterek is elveszhetnek. Még szerencse, hogy az első rossz választ a program nem bünteti.

A második program az ionos kémiai kötéssel kapcsolatos ismeretek tanítására és kikérde-zésre szolgál. Három feladatcsoporton kell sike-resen átjutni, hogy a vájt fülűek hozzájussanak a jutalomjáték lehetőségéhez. Azt most kivéte-lesen nem csodálom, hogy el tudtam jutni a ju-talomjátékig. Besegítettek előző tanulmányai-m. Mégis úgy vélem, hogy a program a témá-ban járatlan tanulókat is sikerrel vezeti rá a he-lyes megoldásokra, úgyhogy sokan eljuthatnak a jutalomjátékig.

A szerzők ügyes fogásokkal buzdítják a ta-nulni szándékozót. (Nem egészen illik ebbe a képbe a „Látom, ezt tudod!” meghökentető megjegyzés.) Az mindenesetre jó, hogy — amint említettem — az első hibás választ még nem bünteti a program. A billentyűzés visz-on-tagságait kiheverve ez lehet a módja a későn észrevert hiba korrigálásának. A buzdítás kö-vetkező fokozata az, hogy ha az első két fe-ladatot elrontjuk, újra lehet próbálkozni. Az ilyen drill módszerrel elég biztonságos tudá-szintre juthat el a nebuló. A programnak egyébként sem stílusa a tanuló letolása a hibák miatt, mint az sok programban látható. Türel-mesen elmagyarázza a megoldáshoz szüksé-ges ismereteket. (A számítógépnek szerencsé-re nincsenek idegei. A programozójának van-nak, de helyesebb, ha nem veszi át a fordító-program hibáuzeneteiben meghonosított le-dorongoló stílust!)

Összességében elmondható, hogy a Juhász Gyula Tanárképző Főiskolán dolgozó szerzőpá-rós jó munkát végzett, ha vannak is észrevéte-leim a programokkal kapcsolatban. Nem sokat markoltak ugyan, de amit megragadtak, azzal jól bántak. Persze sok ilyen programra lenne még szükség a kémia teljes feldolgozásához. Jó szolgálatot tenne, ha a szerzőpáros folytat-ná munkáját ebben a stílusban. Ez elvezethet-ne egy általam negyedik generációs nevez-tett, számítógépi alapokon nyugvó kémiai ok-tatóanyag kialakításához. A két program leg-alábbis ezt sugallta nekem. Ha az említett ta-nanyag megvalósulna, szerepe lehetne az iskolá-n kívüli autodidakta képzésben is. Otthoni számítógépen épp olyan jól használhatók ezek a programok, mint az iskolában.

Zsadányi Pál

### Összefoglaló adatok

**Forgalmazó:** Tudományszervezési és Informatikai Intézet, Kereskedelmi Iroda

**Terméknév:** Kémiai reakciók sebessége. Ionos kémiai kö-tés

**Szerzők:** dr. Berecz Árpádné—Békési József

**Géptípus:** Plus/4

**Hordozó:** kazetta

**Dokumentáció:** két-, illetve egyoldali tömör leírás

**Ár:** 250, illetve 300 forint+ áfa

### Minősítő adatok

<b>Kezelvekészség:</b>	jó
<b>Teljeség:</b>	jó
<b>Dokumentáltság:</b>	közepes
<b>Használhatóság:</b>	kiváló
<b>Ár/teljesítmény:</b>	kiváló
<b>Összbenyomás:</b>	jó

**Dorn Zsuzsanna**  
Tanmenet a VII—VIII. osztály számítástechnika tantárgyának tanításához (Pécs, 1988. Baranya Megyei Pedagógiai Intézet, 37 oldal. Ára: 50,— Ft)

A hiánypótló tanmenetjavaslat hét foglalkozást (21 óra + 4 óra gyakorlat) és a hetedikeseke, tizenhatot (48 óra + 7 óra gyakorlat) és a nyolcadikosok számára. Az oktatási cél a BASIC-programozás elemeinek megismertetése, használatának elsajátítása Sinclair ZX-Spectrum számítógépre alapozva. Tekintve azonban, hogy az alapvető elvek nem kapcsolódnak konkrét géptípushoz, a tanmenet bármely más, az iskolákban elterjedt számítógépre is alkalmazható. Az óravázlatokat eléggé részletes gyakorlati rész egészíti ki. Megrendelhető a kiadónál: 7621 Pécs, Szechenyi tér 9.

**Dr. Poronyi Gábor (szerk.)**  
Számítástechnikai versenyfeladatok I. (Pécs, 1988. Baranya Megyei Pedagógiai Intézet, 72 oldal. Ára: 90,— Ft)

A Nemes Thimér középiskolai számítástechnikai tanulmányi versenyek (1985—1988), a szakmunkástanulók országos tantárgyi tanulmányi versenyei (1986—1988) és a Dunántúli Napló számítástechnikai versenyei (1984—1986) feladatai, a megoldásokat és az értékelés szempontjait tartalmazza. Megrendelhető a kiadónál: 7621 Pécs, Szechenyi tér 9.

## Vitorlás — számítógéppel

A múlt év karácsonyán bocsátották vízre Franciaországban, Le Havre kikötőjében a Lafayette nevű, 400 személyes vitorlás. A világ legnagyobb szélhajtotta hajójának vitorlái számítógépek vezérlik. A hajó hossza 187 méter, öt árbócán a vitorlák együttes felülete 2500 négyzetméter. Az „üsző falu” fedélzetén az utasok a Földközi-tengeren tehetnek majd utazásokat.

## ISDN

A Nyugat-Európában terjedő ISDN (Integrated Services Digital Network) különböző telekommunikációs szolgáltatások — telefon, telex, telefax, videotex stb. — egységes, közös hálózatban való lebonyolítását teszi lehetővé a rendszerben részt vevők között. Ennek előfeltétele, hogy valamennyi szolgáltatás digitalizált legyen. Minden résztvevőnek két csatorna áll rendelkezésére, melyen keresztül két különböző szolgáltatást — akár két különböző partnerrel is — használhat egyidejűleg, 64 kbit/s sebességgel.

Korábban Japán és az USA más utón kívánta a fejlesztéseket végrehajtani. Most jobb a szabványosítás esélyei. A japán posta például liberalizálódott: az NTT magáné lett, melynek részvényeit csak japán állampolgárok birtokolták. Ha erre valóban sor kerül, a szakértők szerint valószínűsíthető legkorábbi időpont is csak 1996 vége.

## Hordozható adatgyűjtő

Az első hazai adatgyűjtők már négy-öt évvel ezelőtt megjelentek a piacon, azonban sehogy sem tudtak széles körben elterjedni. A Trigon Kiszövetveket úgy érzékelték, hogy a csekély mértékű értékesítés alapvető oka ma már nem az adatgyűjtő ára, hanem a hazai piacon lévő adatgyűjtőkhöz adott szolgáltatások szűk köre.



A Psion Organiser II adatgyűjtő

Éberben figyelve a világ adatgyűjtő piacán zajló eseményeket, felfedezték, hogy az e téren egyik legjelentősebb angol cég, a Psion — a szocialista országok közül elsőként — a lengyeleknél januárban elkezdte a népszerű Organiser II típusú gépének a forgalmazását. A Trigon vezetői ott voltak Lengyelországban az ünneplés pillanatán, s azonnal, ott a helyszínen megkezdtek a tárgyalásokat a magyarországi forgalmazásról. Bátor kezdeményezésüket már áprilisban siker koronázta, s megjelentek az Organiser II modell nálunk is forintért.

A 8 bites alapmodell operatív tára 32 kb-ot, amit maximum 2 x 128 kb-ot — egyetlen modulattal cserélhető — RAM vagy EPROM egészíthet ki. Az alappép rendelkezik 2 x 16 karakteres LCD-kijelzővel s 36 gombos, többfunkciós billentyűzettel. Tápellátása egy 9 V-os szabványos elemmel vagy — adapter segítségével — hálózatról történhet. Programnyelve BASIC szintű, azonban a gép nagy előnye, hogy IBM PC-n vagy Macintosh gépen is le lehet rá programot fejleszteni, amit egy adapteren át konvertálható az Organiser II nyelvére. Az alappéphez csatlakoztatható kiegészítők:

- vonalkódolvasó ceruza;
- mágneskártya-olvasó;
- EPROM-másoló és -törölő készülék.

Ezek közül különösen a vonalkódolvasó készüléknek van nagy gyakorlati jelentősége. Az adatgyűjtőt vonalkóddal ellátott árukat tartalmazó raktárban használni például rendkívül gyorsan, s főképpen hibamentesen lehet leltározni. Az adatgyűjtő használati területei igen sokrétűek. Például alkalmazható erdőgazdálkodásban a teljesítmény mérésére, a villanyórák leolvasására, biztosítási ügynököknél a megfelelő árjairól megtérlelés stb.

A várható gyors hazai elterjedés egyik záloga az elérhető ár: maga az alappép csupán 29

ezer forint. Aki alkalmazni szeretné, annak cél-szerű beszerezni — több, adatgyűjtésre használt tenyérnyi alappép mellé — professzionális géphez való csatlakozást s a szükséges professzionális kiegészítőket. Az így létrehozott, egyetlen központi konfiguráció ára százezer forint körüli van.

## Óvodában

Valóban a XX. században járunk: számítástechnikát oktat a Zélicségben az óvó néni. A Dráva menti Dobsza körzeti általános iskolájában 11 falu gyermekeit oktatják, most már számítástechnikára is, a tanterv szerint kilenc éven át. A két esztendőre kezdődött képzést az Országos Pedagógiai Intézet számítástechnikai programirodája segíti. Az oktatás az óvodában kezdődik: a nagycsoportos gyerekek hetente egy órát ismerkednek a számítógéppel. Az általános iskola alsó tagozatában a számítástechnikai órásszám heti kettő. A felső tagozatos gyerekek már dönthetnek, hogy ismereteiket — heti kettő-négy órában — melyik ágon fejlesszék tovább: általános informatikát tanuljanak-e vagy számítástechnikát, avagy a számítástechnika alkalmazását. A tagozatot elvégző gyerekek az általános iskolai bizonyítvánnyal együtt számítástechnikai asszisztens oklevelet kapnak. Később megteremtik rá a lehetőséget, hogy az asszisztensek egy 60 óras tanfolyam elvégzésével számítógépes operátori oklevelet szerezzenek.

## Suprenum

A bonyolult tudományos-műszaki problémák megoldására olyan szuperszámítógépeket fejlesztettek, amelyek részmuveletek párhuzamos elvégzésére képesek. Az NSZK-beli Suprenum-projekt célja a numerikus feladatok megoldásának megkönnyítése a párhuzamos számítás módszerével. Tipikus alkalmazási terület az időjárás előrejelzése, ahol a légkör különböző pontjait leíró, mindig azonos szerkezetű adatok csoportját bonyolult egyenletekbe kell behelyettesíteni, hogy az időjárás alakulása kiszámítható legyen.

Az előrejelzés területeit részreke bontják, melyek mindegyikét egy-egy csomóponti számítógép önállóan kezeli. A fő kérdés eldöntése után a processzorok közötti adatáramlás megoldása jelenti. A Suprenum nem kizárólag hardverprojekt: az operációs rendszer, a programozási nyelvek és az alkalmazói programok olyan együttesére van szükség, mely a gép lehetőségeit optimálisan kihasználja. A kézzelfogható első eredményt a 9 csomóponti számítógéppel működő prototípus jelenti. A tervek szerint még az idén elkészül a 256 ilyen csomóponti számítógéppel működő rendszer.

## Műemlékek

A Veszprém Megyei Tanács több építész-csoport közreműködésével elkészítette a megye műemlékeinek számítógépes kataszterét: 1099 védettséget élvező műemlékről, műemlék jellegű és városképi jelentőségű épületről készült leírás. Az első feldolgozások eredményeként kirajzolódott a műemlékek állapota: a védett épületállományból 524-et ítélték jó karban levőnek, 343-at közepes állapotúnak, a többi rossz állapotban van. Az utóbbiak egy része tanács kezelésű, más részüket mezőgazdasági üzemek birtokolták és többnyire a rendeltetés-merőben ellentétes hasznosítással tönkre is teszik. A számítógépes nyilvántartás megkönnyíti az építményekkel kapcsolatos hatósági feladatokat, ezért minden olyan helyi tanács megkapja, amely építési hatósági jogkörrel bír.



## Szemvizsgálat

A prágai katonai kórház orvosa, Ján Lesták által kifejlesztett számítógépes műszerrel vizsgálják a páciensek szemét. A beteg szemhéja alá apró fémlemez ütletnek be, amely a számítógéphez kapcsolódik. Egy képernyő rövid ideig tartó nézése után a műszer — egy diógram elkészítésével — nagy pontossággal, a beteg szubjektív érzéseinek teljes figyelme kivül hagyásával állapítja meg a kóristém. A diagnosztikai eszköz alkalmas a szemdaganat kezdeti stádiumának felismerésére is, amire eddig nem volt mód.

## Dohányzás

A mai mikroelektronikai eszközöket úgy kell gyártani, hogy még a 1 mikrométernél kisebb paraméterek se zavarjanak. Ezért az Egyesült Államokban több kutató azt vizsgálta, hogy miként tehetnék meg tisztábbá a levegőt a nagyon tisztán tartott helyiségekben, amelyekben a mikroelektronikai eszközök készülnek.

Megállapították, hogy a dohányosok még több órával a legutolsó cigaretta elszívása után is negyszerver annyit szennyezányagot lehetnek ki — főképpen nyolcszempékben és az egyik elpusztult sejtejeinek csomóiként —, mint akik

nem dohányoznak. Ezek a mikroszkopikus szennyezők a kutatók által kiválót egyetlen maszkkal sem tarthatók távol a megmunkálendő tárgytól. A szennyezőssébb erőd sejeit tehát pusztán azáltal is csökkenthető, hogy a kéynes műveletet nem dohányzó munkásokkal végeztetik.

E felismerések bírtakban most azt vizsgálják, hogy mekkora szennyezést okoz a dolgozó ételle, a fogkrémje, vagy a munkahelyiségben hordott ruhamezők mosásának módja. A kutatók szerint különösen veszélyes a baktériumok szennyező hatása, mert sok bennük a mikroóramköröket alkotó félvezetőkre nagyon ártalmas nátrium és kálium.

# ADOK ÉRDEKESSÉGEK

Ebben a rovatban rövid, zövegese, a mikroinformatikai kapcsolatos híreket közlünk. A díjazásban korábban megnevezett a következők közül a legfontosabbakat sorolunk fel: (60 Karaktér) 10 Ft, 160-karaktér az első sor 50,- Ft, minden további sor 20,- Ft. Kérjük, hogy a hirdetés díját az ÁHSTI 07/1. korr. fiókjában (218-9055) vezetett 500-8000 rendezvényváltó-létre vagy 1054 Bp., Bátorfi u. 16. címre fizessék be, rózsaszínű postautalványon (jelölve, hogy a hirdetésért), a befizetés után a díjat az ÁHSTI 07/1. korr. fiókjába a hirdetéshez, hirdetéslehetőségi a szerkesztőség címére várjuk (1373 Bp., Pf.: 433). Az ÁHSTI tagjai továbbra is kedvezményesen hirdethetnek (az első sor ingyenes), de kérjük, hogy adják meg tagjai számukat. Azokat a hirdetéslehetőségeket, amelyek a hónap végén visszatérhetők, a lapunkban.

### ADOK

**Amiga** programok műrészében, 67 cm átmérőjű ITT részén tevényt beépített digitális teletext-előadó. /Nyugtalan olvasószerelem!!!/ Erdőköld-10. 8-21 óra között a 848-47-es telefonon.

**Amiga 500-as (d.)** sok programmal, 4) Philips színes monitor, Atari 800K + floppy eladó. Tel.: 42/18-632  
**Atari 500BT** (512 k) színtelítőpépet 3,5-os lemezzel (360 k), kerféri eladón, illetve megveszve 64 korrál-garanciaúrral. Jentkék Zsolt, Budapest, Szőlő u. 76. IV/20. 1032

**C16-es pépet**, 1. jóstykál, magndval, kb 70 játépprogram, azkirodalommal eladom. Ár: 12 000 Ft. Erdőköld-10. 641-568-as telefonszámra lehet (800) Zoltán.

**C Plus/A-es pépet** 2 jóstykál, magndval, kb 150 játépprogram, szakirodalmat, 1000 db felhívásidőt, 1000 db levelet lehet. Margittay Zoltán, Blauziskola, Dózsa u. 2. 3933

**C Plus/A-hez fényezésű eladó.** Keresek: nyak tervezőt, Erdőköld-10. harvort kérlek. Bonyhád, Bonyhád, Kossuth u. 57. 7150. Tel.: 74/31-1

**C Plus/A-re 12 db játépprogramot** eladó. Kertész. Válaszlevelet kérek! Rádió Ferenc, Budapest, Ruda u. 2. 1208  
**C Plus/A-re** 1000 db felhívásidőt és játépprogram lemez, kazettát, átlag 16 Ft/db arú eladó. Kérésre listát küldök. Szavári Gábor, Lenti, Petőfi u. 33. 8960  
**C Plus/A-re** hangjátékot, 2500 lemezű, Reser-Super Reser kapcsoló, C64-es kazettás egyfedre hardveres elsőso készlet eladó. Válaszborítékot írszertó küldök. Tócsányi László, Szeged, Rózsa F. sq. 128/a. 6726

**C64-es lemezok eladók** 200 Ft/db. super programokkal. Kérésre listát küldök. 1351-es nrgr. Citzen 120 + interfész, magnd 4000, 35 000, 2000 Ft) eladók. Németh Ferenc, Budapest, Nagyvány u. 8/a. 1184

**C64-es és C Plus/A-re játépprogramok** eladó. Válaszborítékot listát küldök. Barát Imre, Győr, dr. Pogány Irén u. 22. 9024  
**C64-es 1987/88-as** programokat adok (20 Ft/db), az csomagok kazetták és lemezek. Ifj. Varga Péter, Nyíregyháza, Dvoda u. 49. 4400  
**C64 programok eladók** (15-30 Ft). Listát küldök. Mérti László, Orosháza, Sándorfi u. 12. 22-026/2

**Comodore 64 tulajdonos** vállalkot, magndszámlák! C64-hez CP/M cartridge

(Digital Research gyártmányú), most 50k-os árcénpépettel, csak 10 000 Ft. Ürális műfjver hátter; a 280 programmal megvalósított kizárólagos a gépbe adott. Erdőköld-10. Lakatos Péter, Munkács II. Klapka Gy. u. 36. 3524. Tel.: 46/64-417

**CI16-es színtelítőpp** magndval, programokkal, kézi vezérléssel 10 000 Ft-ig. Erdőköld-10. Lehet: Bekeskovács Sándor, Nagykanizsa, Kiscsokva u. 39. 9485

**C128 VC 1541, VCS2000** eladó. Erdőköld-10. Gyevyer Tamás, Kiskunhalas, Ady E. u. 17. 6400

**Comodore 128 és 1571-es** floppy eladó. Kesztes a ROKKONLITR 5-3 programot. Kesztes Ferenc, Eger, Belsőváros u. 7. 3300

**Citizen 120-C** 64 megkezelésű leírás-eladó. 300 Ft csökken vagy utóváltó. László István, Dunajváros, Hengerész u. 5. 2400

**Datatesseks figyelni!** IC-es másodikok eladók! (2 Datateset közti adatok) lehet még a kézen tartók között! 100 talas, Székességhévír, Vöröshégy ut. 62. 8000

**Eladom** a Mikrocsztópog Magán és a BT-ET (Egyesített Szivolyagú) bekiötve. Telefon: 330/60/43

**Eladó** a Computerworld Számítástechnika eddig megjelent összes száma; Plus/A programokat lemezre cserelek. Kálmán Albert, Eger, Kossuth u. 31. 1171/1. 3300

**Eladó ZX Interfész 1** - a microdrájv + 4 cartridge leírásokkal. Erdőköld-10. Polgár József, Kesztemet, Balaton u. 20. fsz. 30. 6000. Tel.: 76/47-263

**Eladó 80-k** Spectrum jóstykál, 20. felsz. 2-vel és több száz programmal. Kerecsényi Tamás, Pécs, Zauzannya u. 10. 7632

**Eladó egy** alig használt Comodore 1351-es Mouse. Erdőköld-10. 04 óra után telefonon: 688-558

**Enterprise programokat** adok-veszek, és hozzájárva csempereket keresek. Kb 150 db jóstykál programmal rendelkezem. Sala István, Budapest, Hárfa sétány 17. 5/30. 1203

**Enterprise programokat** adok, csemperek, olcsón 20-30 Ft/db. rövid határidő alatt. Németh Antal, Bonyhád-Majos, VIII. u. 26. 7187

**Enterprise, Comodore Plus/A, Spectrum** programokat adok-veszek-csemperek között és lemezek. Csere esetén listát kérek, vásárlás esetén válaszborítékot esetén listát küldök. Kb 4900 programom van. Ifj. Kovács Ignác, Siklód, Pf.: 105. 7800

**Enterprise programokat** adok, csemperek. Válaszborítékot listát küldök. Cím: Majos, II. u. 17. 7187

**Enterprise programok** olcsón eladók. Listát küldök, válaszborítékot kérek. Bányai Gabna, Nyíregyháza, Tanácskörtárság tér 3. 4400

hardver dokumentáció, 20 db megismerő programokkal + dokumentáció. Miron Gyula, Debrecen, Karácsony Gy. u. 9. IV/16. 6290

**I.C.M. - THE NAME OF QUALITY** I.C.M. - a legjobb és leggyorsabb eladó, programfejlesztés: Comodore 64 (dísz/hazetta); kb 150 program havonta; Amiga több mint 30 címek a legjobb programokkal havonta; Atari 31: több mint 30 új címek és több mint 30 program havonta

**IBM** biztosítja az Ön ellátását a legjobb programokkal a felsorolt géptípusokhoz. Levelezek angolul vagy magyarul. Milonovich Slobodan, 19000 Zagreb, Szélesség út 11. C/I-39, Jugoszlávia. Tel: 019-21010

**Minden típusú személyi számítógéphez** sokasáron van vezérlő hardver, szoftver eladó, olcsón. T. Nagy Zsolt, Kesztemet, Fehér u. 20. 6400

**Primo SSB**-hez 3 k-es komplett felhasználói programcsomag ártétele. Egyéb Epron digitális mestertől vagy lemezzel 24 órán belül. Vádrész u. Horváth László: 583-54 (napközben)

**Sinclair Spectrum játépprogramok** eladók (mindig a legjobb szoftverek). Sashegyi Zsolt, Pécs, Kalánfalvi út 45. 11/10. 7625

**SPECCY-00S**, garantáltan a legjobb Spectrum díszköz interfész + 5'1/4" MM floppy eladó! Irányár 10 000 Ft + 4000 Ft. Erdőköld-10. Lehet: Ivelován Kardos Balázs, Budapest, Szécsa A. u. 9/A. 1021 vagy tel.: 658-814

**Spectrum, C Plus/A és C16-es programok** olcsónál 15-20 százalékos eladók. 10 Ft/db. Válaszborítékot listát küldök. Nagy Zoltán, Székességhévír, Szepesi u. 42. 8000

**TV-Computer programokat** adok-veszek-közvetlenül, jóstykál, programokkal (kazettán). Irányár 10 000 Ft. Erdőköld-10. Lehet telefonon: 872-789, ifj. Vass Gáborral

**ZX-Spectrum** 2 db Microdrive, 2 db bootkormány csatló, egy Sokolova GP 50 nyomatkozó, könyvek, felhasználói programok és nyelvek leírásai eladók. Lista, alárnyaló. Dr. Rózsa Sándor, Jászberény, Ráczné sétány 5. 3400

**ZX-Spectrum 600** db válogatott programmal. Kempton interfészzel, könyvekkel, válogatott C64 és 1541-11. floppy olcsón lemezzel eladó. Szivoczka Ernő, Szeged-Szőreg, Szerb u. 30. 6771. Tel.: 55-061

### VEZSEK

**Amigához** szakirodalmat, újságot, verneket, információk, programok cseréje! Bárcos Ferenc, Kalocsa, Miskolc út 20. 6300

**C Plus/A-re** keresem az Ausztrópépet, Unicomp, Compacker nevű programokat. Vízvári Balázs, Micsony, Kossuth út 81. 3700

**Keresek** megvetésre 10 db Ferranti/UA 1HD5E308 IC-2. Szabó Zsolt, Mécse, Komáromi út 3/a. 3871

**Program 128-as** vezérlő. Telefon 16-19 óra között 126-506  
**Vennek Atari 800 XI**-hez magndt és programokat kazettán, vagy cserébe kívánom szúrni elektronikus készleteket. Tel.: 225-026/2. Kovács István, Kertai, Rákóczi u. 38. 1173

### CSELEK

**Amiga 500 tulajdonosok** keresek információt és programcsere. Listát kérek. Kúldök. Bárcos Ferenc, Kalocsa, Miskolc út 20. 6300

**Amiga** Comodore Amiga tulajdonosokkal leveleznek (felhasználói- és játépprogram cseré céljából). Keresek bármilyen német nyelvű játépprogramot! Válaszokat, programlistákat a következő címre várjuk: Frank Attila, Domaszék, Petőfi u. 5/30. 6781

**Comodore Amiga 500 típusú** számítógép tulajdonosokkal leveleznek. Rágo István, Budapest, Dózsa u. 64. 1121

**C16, C Plus/A programokat** cseréllek lemezek és kazettán. Címünk: Kiss Kálmán, Debrecen, Símanyi u. 17. 4020; B. D. Róbert, Debrecen, Ságvári E. u. 8. 4025. Tel.: 52/23-313

**C16, C Plus/A és C64-es** programjaimat királom fel cserére, szálagon és lemezen fleg oktatási és játépprogramok értékesítésére. Lehetőség listát kérek! Várom a régi csempereket! Jelentkezést! Itt: Hiba György, Szeged, Símanyi u. 1/A. 6724. Tel.: 62/30-496 (17 dru után) Gábor

**C16, C Plus/A programokat** cseréllek. Listát kérek és küldök. Simon László, Szolnok, Széchenyi u. 7. 11/7. 9000

**C Plus/A-es** szivolyagos programokat cseréllek-veszek. Válaszlevelet listával kérek. Rácz Ferenc, Nyíregyháza, Liget u. 1. 4400

**Comodore Plus/A-es** programokat cseréllek. 300 programom van. Bertók Gábor, Budapest, Naszd u. 1. 1202. Tel.: 277-032 (est)

**Comodore 64-es** játépprogramokat cseréllek. Cseréltem 1600 db bulvárúradó program. BASIC és hasonló prg-k nem érdekelnek! Válaszokat csak leveletben, listával kérek! Úgyanott eladó! Duhokhot 13. 70yball. Nem használta! Ár: 1600 Ft. Egri Tamás, Hévíszentandrási, Fecsirta u. 6. 5561

**C64-es** programokat cseréllek kazettán. Listát kérek. Bányák Mihály, Mosonmagyaróvár, Széchenyi u. 19. 11/8. 9000

**C64-es** programokat cseréllek. Válaszlevelet listával kérek. Illés István, Nyíregyháza, Liget u. 8. 4400

**C64-es programokat** cseréllek lemezek és kazettán. Listát kérek! Hagasztó István, Mezőgyehény, Kálcsy u. 13. 9820

**Comodore 64-es** játépprogramokat cseréllek lemezek. Kerecsényi a California Games nevű programot. Ne lehet azoké! Kérlek: 81. 210-es számra. E megkezdés után válaszok. Varga Imre 6/b oszt. tanuló, Városföld, Béke u. 38. 6033

**Enterprise programcsere.** Listát kérek! Varga András, Veszprém, Arany u. 33. 8200

**Enterprise programcsere,** olcsó eladás. Lázár István, Mohács, Dózsa u. 8. 7000 vagy Újváros, Mohács, Kálcsy u. 6

**Enterprise programokat** cseréllek lemezekre. Listát kérek és válaszborítékot ellenében listát küldök! Szóles program-válaszokat. Hévír Péter, Eger, Faiskola u. 8. 17/6. 3300

**Enterprise programokat** cseréllek, listát kérek! Rózy György, Tatabánya, Budai, Lévai út 31. 8/7/9. 2043

**Enterprise programokat** cseréllek. Stumpf János, Budapest, tel.: 559-673 (est)

Er a rovatunk KODEX 2000 szívesgarnitúrával készült.

# Az Olvasó írja

Válószínű, hogy MIKROKAGAZIN nem volt, így nem is szeretném róla tudomást.

Ezért a MIKROVILÁG-ot vettem rendszeresen.

Egyik kollégám 1988 szeptemberében behozott egy MIKROKAGAZIN-t. Megnéztem és azonnal „átnyergeltem”, sőt erre az évre elő is fizettem a MIKRO-ra.

Úgy érzem, ez a lap jobban szolgálja a gép és ember közötti kapcsolatot közelítését, mint a többi.

Nagyon örülök a hardverismertetéseknek és a felhasználói rutinoknak. Magam is szeretném a programozást „kitanulni”, ehhez kérem segítségüket.

Úgy érzem, a most már népes ENTERPRISE-tábor nevében kérhetem, hogy indítsanak cikksorozatokat a gépi kódú programozásról, az ENTERPRISE adottságait figyelembe véve.

Köszönjük a levelet, az Enterprise-zal kapcsolatos javaslatát a rovatszerkesztőnek továbbítottam.

## Blaskó István, Mosonmagyaróvár

Az  $\mu$ -ban olvastam, hogy fantáziát látnának egy levelezőklub beindulásában, de eddig még mindenkit elémított a sok papirunkna, amivel ez a munka járna. Amiért jelentkeznék — sajnos nem vagyok egy „kreatív típus” programozás terén, de talán a levelezőklub keretében szerény munkámmal sikerülne egy kevéske segítséget nyújtani azoknak, akikben több tehetség, ambíció van. Igaz, én Mosonmagyaróváron lakom, de hetente 1 napot tudnék szánni e jelentkezésre, ha ez szükséges.

Nagyon sokszor üzentem már ebben a rovatban a vállalkozó olvasónak, hogy szívesen vesznék mindenféle ötletet, javaslatot, együttműködési elképzelést, ha pontosan tudjuk, hogy mennyiben számíthatunk olvasónk közreműködésére. Ezért kérem mindazokat, akik velünk akarnak dolgozni, azok ne csak a „mit”, hanem a „hogyan”-t is írják meg, különben nem tudjuk elképzeléseiket a rövid vagy hosszú távú terveinkbe beiktatni.

Szívélyes üdvözléssel:

Kovács Győző

**Ebben a hónapban ismét a szerzői joggal, a programok forgalmazásával, a cserével, vétellel-eladással kapcsolatos problémákról szóló levelek között válogattunk.**

## Lakatos Péter

Az 1989/2-es számban Pákozdy Pál kifejtette a véleményét a játékc- és felhasználói programleírások közléséről. Szerintem nem leírásokat, hanem térképeket kellene megjelentetni az újságban. Abban sem értek egyet Pákozdy Pállal, hogy az ilyenekért nem kellene fizetni az újságnak. Ugyanis ha ezt tenné, akkor:

1. Mindenki a Spectrum-világnak küldené be.

2. Azzal a jelszóval, hogy még a bélyeg ára sem térül meg, be sem küldenének térképeket. (Ennek az újságnak.)

Visszatérve levelem elejére, a leírásokat azért nem tartom célszerűnek, mert rendszerint minden számítógéphez megjelenik egy olyan könyv, amely leírásokkal foglalkozik.

Más. Nagyon jó ötletnek tartom azt, hogy az ENTERPRISE-nak egy külön rovatot hoztak létre. Még jobban örülök annak, hogy az EXOS 2.1 c. könyv alapján példaprogramokat is közöl néha a MAGAZIN. Viszont az még jobb lenne, ha alaposabb lenne a leírás hozzájuk.

Olvasónk bizonyára észrevették, hogy lassan elmúlik az a világ, amikor a Mikromagazinhoz hasonló újságok oldalas utasítástáradák közölnek, amelyekből mindenki bepötyögtheti a számítógépébe a kiválasztott programot. Valóban arra törekszünk, hogy lehetőleg rövid példaprogramokat, programozási és technológiai fogásokat közöljünk a korábban gyakorta hozott hosszú program helyett. Ami a szerzőknek való fizetését illeti, véleményem szerint a szerzőket a szerzői díj megilleti.

## Kovács Péter, Nagyvenyim

Commodore Plus/4 tulajdonos vagyok. A gépet nemrég kaptam, ezért még nincsen sok programom (kb. 40). És ezért nem tudom gyarapítani programjaim számát. Mint ön is tudja, sokan hirdetnek programcsere ügyében. Volt rá példa, hogy irtam egy hirdetőt, de ő visszazárt, miszerint a programjaim mind megvan nekik és ő így nem cserél. De ELAD programokat: darabját 20 (!) forintért.

Kérdésem én: Hát csak pénzért lehet programot szerezni?

Hát nem is tudom, hogy hol kezdjem, hiszen a programcsereéről, a programeladásokról, a szerzői jogról már eddig is sokat beszéltek. Állandóan visszatérő problémánk, hogy szabad-e, erkölcsös-e programokat cserélni, hiszen ily módon minden csere a program szerzőjét károsítja. Kétségkívül a programhoz jutás legbiztonságosabb módja, ha valaki bemegy a boltba és a programot megveszi, ekkor egészen biztosan hibátlan árut kap, probléma esetén a vevő a garanciális jogait érvényesítheti. Elfogadott, de jogilag egyáltalán nem tisztázott módja a programok megszerzésének a csere. A csere során a partnerek látszólag nem jutnak anyagi előnyökhöz, azonban ez teljes mértékben nem igaz, hiszen a programkészletük mégis csak gyarapodik. A csere az egész világon elterjedt programszerzési forma, nemrég

hallottam Ausztriában, hogy a játéckprogramoknak cca 60%-a így cserél gazdát. A programok forgalmazásának a legbiztonságosabb módja — véleményem szerint —, ha valaki a cserével szerzett programot sokszoroztatja és pénzzért árulja. Ezt a tevékenységet minden tisztességes amatőrnek el kell ítélni. Ami pedig a hirdetőt illeti, ha a saját maga által készített vagy mástól forgalmazással célul vásárolt programokat árul, akkor a vásárlás ellen semmi kifogása nem lehet, ha károlmasolatokat, akkor annál inkább. Ami pedig az utolsó kérdését illeti, arra csak azt tudom mondani, hogy „bizony, így kellene szerezni a programokat!”

## Dr. Halmágyi Győző, Nagyszentjános

Foglalkozásomra nézve körzeti orvos vagyok. Régióta foglalkoztam a gondolattal, hogy a munkám során elvégzendő rutinteg adminisztrációs feladathoz miként tudnék mikroszemélygépet igénybe venni. Sajnos egy minden feladatot kielégítő PC és a hozzávaló „testre szabott” szoftver elérése az egészségügy jelenlegi anyagi lehetőségei mellett csak álom. Ismerve a C 64-es gép lehetőségeit, megalkuvással, de jól tudnám használni. Lehetősegem nyílt, hogy szinte ingyen kapjak egy C 64-es alapgépet, két lemezmaghajtóval, monitorral, nyomtatóval. Tehát a hardver része a rendszernek adott lenne. Találtam egy kollégát, aki társával, saját tapasztalatai alapján, összeállított egy gyakorlatban is kipróbált és bevált nyilvántartási programot. Át is adná szívesen. Gondom az, hogy a program a Precision Software Inc. által kifejlesztett „Superbase 64”-re épül. Próbáltam itt Győrben a Novotrade szaküzletben, a SZÜV-nél, Budapesten a Novotrade Aruházban, a SZÜV-nél, a SZÁMALK-nál, az ECONORG-nál, de mindenütt azt a választ kaptam, hogy C 64 programokkal nem foglalkoznak (?). Sikerült ugyan megszereznem a Novotrade kiadásában (!) megjelent Áts: Superbase 64 című felhasználói kézikönyvet, de a programra még csak tippet se kaptam, hogy hol lehetne beszerezni.

Tisztelettel kérem, mivel önben van utolsó reményem, segítsen, hol lehet Magyarországon hozzájutni a Superbase 64 adatbázisrendszerhez.

Ha ön sem tud segíteni (mivel hivatalosan külföldről nem vásárolhatom meg a rendelőt, és leltárba kell venni), kénytelen leszek törvénysértő módon valakitől lemásolni. Törvénysértő (és valószínűleg naiv) ember vagyok, e lehetőséget szeretném legutoljára hagyni.

Megpróbálom segíteni, ezzel kapcsolatban Halmágyi doktor urat levelben értesítem. A levelet azért közlöm, mert válasz a megelőző kettőre, még ha problémák vannak is, akkor is a legális utat kell szerintem követni és nem csak a „naiv” embereknek.

## Szkárosi Gyula, Budapest

1987 júniusában vásároltam ENTERPRISE számítógépet. A vásárlás napján felderítő útra indultam, számítógépes újságokat keresni. Háromfélét is kaptam, köztük MIKROVILÁGOT.

## NÁLUNK MÉG KAPHATÓ

Első könyvem a mikróról  
30,— Ft

Első könyvem a programról  
30,— Ft



Kereskedelmi Iroda  
1145 Budapest,  
Szugló u. 9—15.  
Telefon: 642-000/176-os,  
177-es mellék

ISKOLÁKNAK OKTATÓKNAK SZAKKÖRÖKNEK

**CSÜV**  
Szekszárdi Számítóközpont:

## Komplex számítógépes szolgáltatásaink:

- Számítástechnikai kutatási feladatok
- Mikrogeptől nagygépig — hagyományos és interaktív számítógépes felhasználói rendszerek kidolgozása
- Számítógépes rendszerek üzemeltetése
- Kiadványszerkesztési szolgáltatás
- Mikroszámítógépek értékesítése, hálózatok kialakítása
- Számítástechnikai alkalmazásokhoz szükséges segédeszközök forgalmazása
- Számítástechnikai berendezések szervize
- Szaktanácsadás
- Oktatás

## Vállalkozási irodák:

- BUDAPESTI VÁLLALKOZÁSI IRODA  
1132 Budapest XIII., Kresz Géza u. 44—46.  
Telefon: 209-816
- FONYÓDI VÁLLALKOZÁSI IRODA  
8640 Fonyód, Ady E. u. 29.  
Telefon: 84-60-342
- COMPUTER—M ÜGYFÉLSZOLGÁLATI IRODA  
(KERESKEDELMI RÉSZLEG)  
7100 Szekszárd, Wesselényi u. 15.  
Telefon: 74-16-822

Várjuk tisztelt ügyfeleink érdeklődését:

ALISCA SZOFTVERHÁZ  
7100 Szekszárd, Wesselényi u. 15.  
Telefon: 74-16-822  
Telex: 14-363



# Vegyészeti csemegék

